**שם המורה:** יהודה אבני

**מגישים:** עומרי יעקובי, אביב פרדו

**ת.ז:** 315229922 , 211434915

**בית ספר:** תיכון חקלאי פרדס חנה - ברנקו וויס

**כיתה:** יב'8

**תאריך:** 26.4.2018

**רקע לפרויקט**

הקדמה

הפרויקט שלנו הוא בניית ואבטחת מערכת העברת נתונים בתקשורת בין שרת ללקוח בפרוטוקול TCP. המשתמש נדרש לפתוח משתמש ולאמת אותו על ידי קוד הנשלח בדואר אלקטרוני. לאחר ההרשמה המשתמש יכול להתחבר למשתמש שלו, ובכל כניסה "להפקיד" ו"למשוך" כסף דרך התכנה. השרת מאזין לבקשות בפורט מסוים ברשת המקומית, וברגע שנכנסת בקשה מאחד הלקוחות, מתחילה תקשורת מאובטחת בין הלקוח לשרת. השרת מחובר לבסיס נתונים, התקשורת עוברת מהלקוח לשרת לבסיס הנתונים ואז חזרה אל השרת ואל הלקוח. הלקוח אינו יכול לגשת אל בסיס הנתונים. הרעיון הכללי של הפרויקט הוא לדמות כספומט וירטואלי שבטוח באותה מידה כמו כספומט פיזי ואם לא יותר.

מדוע בחרנו בנושא זה

בחרנו בנושא זה מכמה סיבות שונות:

1. הנושא של אבטחת מידע משך אותנו. הרעיון שהיישום שלנו הוא כספומט מדגים בצורה הטובה ביותר למה צריך אבטחת מידע. אחד הדברים החשובים ביותר של אנשים הוא הרכוש (כסף) שלהם, הם מעוניינים בהגנה עליו בכל מחיר ולכן אבטחת המידע בפרויקט הופכת למטרה המרכזית. הרצון לחפש כל הזמן שיטות חדשות כיצד להגן על הנתונים בצורה המרבית סחף אותנו כל פעם מחדש. בימים אלו עם פריצת המידע לפייסבוק הנושא הופך להיות רלוונטי מתמיד, ואנו מגלים כי אין איש הרוצה כי פרטיו או חשבונו יהיו מוצגים לציבור הרחב. חברות רבות כיום מרחיבות את תחום אבטחת המידע בעקבות הרלוונטיות של הנושא.
2. תקשורת בין מחשבים הוא נושא מאוד רלוונטי היום, עניין אותנו כיצד מחשב אחד יודע את כוונתו של מחשב אחר? בעקבות הסקרנות הרבה בתקשורת בין מחשבים החלטנו לחקור ולבדוק כיצד הדבר עובד וכיצד ניתן ליישם אותו בהתאם לפרמטרים שלנו (מהירות, אבטחת מידע וכו').
3. עבודה עם בסיס נתונים עניינה אותנו, במיוחד כתיבה בשפת SQL. השתמשנו בפרויקט כמסגרת שבה - תוך כדי הכנת הפרויקט עצמו, נוכל ללמוד זאת.

הבעיות שהתכנה פותרת

1. פתרון נוח ובטוח לשליטה, ניהול ומעקב אחר הוצאות והכנסות.
2. המערכת מעניקה למשתמש אפשרות להשתמש בכל הכלים של כספומט מביתו ללא טרחה, באופן וירטואלי לגמרי ומאובטח ברמה של כספומט פיזי ואולי אפילו יותר.
3. מאפשרת העברת כספים באופן וירטואלי בלי קישור לבנק.

משתמשים פוטנציאלים

חברות בנק

כל אדם/חברה/גוף שרוצה לנהל את כספו

תחומים

אבטחת מידע

תקשורת בין מחשבים

בסיס נתונים

SMTP

**מטרות הפרויקט**

מה הפרויקט אמור לבצע:

התוצר הסופי אמור לאפשר למשתמשים להשתמש בתכנה לצורך הפקדה ומשיכה של כסף מחשבונם, בצורה מאובטחת, מהירה וחלקה.

מטרות מרכזיות

לאפשר יצירת קשר בין לקוחות לשרת באופן קבוע, ולאחר מכן זיהוי הלקוח ובמידת הצורך, הרשמת לקוח חדש והכנסתו לבסיס הנתונים. לפני שלמשתמש תהיה אפשרות להתחבר לחשבונו תתחרש בדיקה שהמשתמש אינו רובוט על מנת לשמור על המערכת מאובטחת מפני בוטים.  
אבטחת החשבונות על ידי שימוש בדואר אלקטרוני אשר מקושר לחשבונם ועל ידו הם מאשרים את חשבונם.

מטרה חשובה היא הצפנת התקשורת לכל אורכה על מנת למנוע חדירה של צד שלישי. כל מידע שמועבר בין השרת לכל אחד מהלקוחות מוצפן עם מפתח נפרד לכל לקוח, ומפתח נפרד לשרת בתקשורת עם כל אחד מהלקוחות השונים, כך שאין דרך שצד שלישי כלשהו יוכל לפענח את המידע שעובר ברשת.

שפות

הפרויקט שלנו עושה שימוש בעיקרו בשפת C# כאשר צד לקוח רשום כולו בC# ובצד שרת יש גם שימוש בשפת SQL בכדי לקשר בין הנתונים שמקבל השרת לבין בסיס הנתונים.

סביבת עבודה

הפרויקט כולו נעשה ב Visual Studio. הן הקוד בC#, הקוד בSQL והחיבור לבסיס הנתונים.

**ניתוח הבעיה האלגוריתמית**

1. ביצוע תקשורת עם מספר לקוחות בו-זמנית.

השרת רשאי לתקשר אך ורק עם לקוח אחד על סוקט מסוים, לכן הוא איננו יכול לתפקד כאשר מחוברים אליו 2 משתמשים או יותר אם הוא עובד רק עם Socket אחד. למערכת שלנו חייב שהשרת ידע לתקשר בו זמנית עם מספר רב של משתמשים.

1. הצפנת המידע נגד צד שלישי:  
   על מנת שצד שלישי אשר "יושב" על החיבור בין הלקוח לשרת לא יוכל להבין את ההודעות אשר עוברות בין שני הצדדים, ועל מנת שלא נצטרך להעביר מידע רגיש כלשהו כדי להתחיל את התקשורת, יש להשתמש בהצפנה א-סימטרית. עם הצפנת המידע צצה עוד בעיה, להמיר את המידע לבייטים, להצפין אותו, לשלוח ולהמיר בחזרה בלי שהמידע ייפגם.
2. יצירת CAPTCHA - בדיקת אנושיות:  
   על מנת שתכנות רובוטיות וזדוניות למיניהן לא יוכלו להציף את השרת או לנסות לפרוץ לחשבונות הלקוחות (BRUTE FORCE), יש לבצע בדיקת CAPTCHA בכל התחברות של לקוח חדש. על בדיקת הCAPTCHA להיווצר בצד השרת כדי שללקוח זדוני לא תהיה את האפשרות לדוג את הטקסט הנכון מהקוד של הלקוח ועליה להיות מועברת בצורה מוצפנת כתמונה. כמו כן על הבדיקה של נכונות הפתרון לCAPTCHA הנשלח מהלקוח לשרת להיות בצד שרת, כדי שהטקסט הנכון לא יועבר בתקשורת בין השרת ללקוח.  
   יש ליצור בכל פעם שנוצרת CAPTCHA טקסט אקראי בעל רווחים משתנים בין התווים וקשקושים אקראיים בתמונה כדי שרק בן אנוש יוכל לפתור אותה.
3. זיהוי הבקשות של הלקוח ותגובות השרת

השרת מאזין כל הזמן ללקוח ולהפך, הבעיה העיקרית של תקשורת זו היא ההבנה ביניהם, לדוגמה כאשר הלקוח רוצה להוציא כסף כיצד השרת יודע שהלקוח רוצה להוציא כסף ולא להכניס כסף. השרת צריך לדעת איזו פעולה הלקוח רוצה לעשות בכל פעם שהוא מקבל ממנו הודעה, גם כן הלקוח צריך לדעת מהי המשמעות של כל הודעה מהשרת.

1. שליחת דואר אלקטרוני  
   הוספת רובד של אבטחה ואימות מתרחשת בעת קישור דואר אלקטרוני לכל משתמש. לכן, בכל הרשמה של משתמש חדש יש על השרת לשלוח הודעת דואר אלקטרוני ללקוח, אשר מכילה קוד שעל המשתמש לשלוח בחזרה לשרת. הבדיקה של נכונות הקוד צריכה להיות בצד השרת.
2. ייעול זמן תגובה של התקשורת  
   בכדי שהיישום יהיה אפשרי ונוח לתפעול צריך לקצר את זמן התגובה של השרת לבקשות הלקוח. בתחילה השרת מתחבר אל בסיס הנתונים והדבר גורם לעיקוב משמעותי בזמן התגובה. בנוסף כאשר השרת מכניס משתמש למערכת עליו קודם כל לבדוק האם שם המשתמש תקין (האם הוא קיים כבר במערכת). לאחר מכן בודק האם ניתן לשלוח דואר אלקטרוני לכתובת הדואר האלקטרונית שהתקבלה ובמידה וניתן שולח לו הודעת דואר אלקטרוני שמכילה קוד סודי ובסוף התהליך מכניס את הלקוח אל בסיס הנתונים, פעולה זו לוקחת זמן רב ויש למצוא דרך לייעל אותה.

**ניתוח הבעיה האלגוריתמית**

1. ביצוע תקשורת עם מספר לקוחות בו-זמנית.  
   - פתיחת אובייקט נפרד לכל לקוח ולכל אובייקט של לקוח פתיחת Socket משלו. כך השרת מסוגל לתקשר עם כמה לקוחות בו-בזמן.
2. הצפנת המידע נגד צד שלישי:  
   הצפנה א-סימטרית - RSA.  
   בתחילת התקשורת הלקוח והשרת מעבירים אחד לשני את המפתחות הציבוריים.  
   כל צד מצפין באמצעות המפתח הציבורי של האחר ומפענח באמצעות המפתח הפרטי שלו.  
   המידע מומר לBase64 String לפני המרתו לבייטים על מנת לשמור על המידע תקין (מניעת corruption).
3. יצירת CAPTCHA - בדיקת אנושיות - בצד שרת ולהעביר אותה ללקוח.  
   על מנת שתכנות רובוטיות וזדוניות למיניהן לא יוכלו להציף את השרת או לנסות לפרוץ לחשבונות הלקוחות (BRUTE FORCE), יש לבצע בדיקת CAPTCHA בכל התחברות של לקוח חדש. על בדיקת הCAPTCHA להיווצר בצד השרת כדי שללקוח זדוני לא תהיה את האפשרות לדוג את הטקסט הנכון מהקוד של הלקוח ועליה להיות מועברת בצורה מוצפנת כתמונה. כמו כן על הבדיקה של נכונות הפתרון לCAPTCHA הנשלח מהלקוח לשרת להיות בצד שרת, כדי שהטקסט הנכון לא יועבר בתקשורת בין השרת ללקוח.  
   יש ליצור בכל פעם שנוצרת CAPTCHA טקסט אקראי בעל רווחים משתנים בין התווים וקשקושים אקראיים בתמונה כדי שרק בן אנוש יוכל לפתור אותה.
4. זיהוי הבקשות של הלקוח ותגובות השרת

הזיהוי מתבצע על פי פקודות מוסכמות שהוחלטו מראש. הפקודות תקפות גם לבקשות הלקוח וגם לתגובות השרת.

1. שליחת דואר אלקטרוני  
   שליחת הדואר האלקטרוני מתבצעת באמצעות Outlook כאשר ישנו שימוש ב SMTP - Simple Mail Transfer Protocol לכתיבת הודעה מOutlook. השרת מייצר קוד - רצף של אותיות באופן אקראי ושולח אותם בגוף ההודעה, לאחר מכן מכניס את הקוד אל תוך בסיס הנתונים בעמודה המתאימה. ללקוח נפתח חלון להכנסת קוד (ישנה אפשרות לקבלת קוד חדש מהשרת). במידה והקוד נכון השרת מוחק את הקוד מבסיס הנתונים במידה ולא, השרת מודיע ללקוח כי טעה ברצף האותיות. לאחר שהכניס קוד נכון הלקוח רשאי להיכנס לחשבונו.
2. ייעול זמן תגובה של התקשורת  
   בכדי לייעל את זמן התחברות השרת לבסיס הנתונים השרת בודק את החיבור עם בסיס הנתונים כבר בתחילת היישום ולא רק כאשר השרת מתבקש להשיג מידע מבסיס הנתונים. תהליך נוסף בו מייעל השרת את הזמן הוא בהתחברות. כבר לאחר קבלת האישור שהשם משתמש תקין שולח השרת בו זמנית ללקוח כי ההתחברות הצליחה ובנוסף שולח לו דואר אלקטרוני ומכניס את המשתמש למערכת.

**יתרונות וחסרונות**

1. ביצוע תקשורת עם מספר לקוחות בו-זמנית
   1. שימוש בחיבור בפרוטוקול TCP-
      1. יתרונות-
         1. חיבור אמין ועקבי בין הלקוח לשרת, כל מידע שיישלח מהלקוח יתקבל בשרת ולהפך.
      2. חסרונות-
         1. חיבור איטי לעומת האופציה השנייה, (Three-Way-Handshake), דורש מעבר של יותר הודעות.
   2. שימוש ב Socket נפרד לכל לקוח
      1. יתרונות-
         1. עבודה עם כמה לקוחות בו זמנית ולמנוע ו"דריסת" Socket.
      2. חסרונות-
         1. יוצר עומס, חוסר יעילות בניצול מירבי של כל Socket.
2. הצפנת המידע נגד צד שלישי
   1. שימוש בRSA-
      1. יתרונות-
         1. כל התקשורת תהיה מאובטחת. פחות סכנות של האזנות מצד שלישי.
         2. לא נדרשת הבנה מעמיקה כיצד מתבצעת ההצפנה
      2. חסרונות-
         1. לא ניתן לדעת מה השרת מעביר ללקוח בכדי לפקח על תגובות השרת.
3. יצירת CAPTCHA - בדיקת אנושיות - בצד שרת ולהעביר אותה ללקוח
   1. העברת תמונה מהשרת ללקוח-
      1. יתרונות-
         1. מונע מעבר של לקוחות ללא בדיקה. במידה והבדיקה בצד לקוח, הלקוח יכול לאשר את הבדיקה בעצמו.
      2. חסרונות-
         1. יצירת מחלקה לשליחת תמונה שהשימוש היחידי שלה הוא ביצירת הcaptcha.
         2. שליחת התמונה ללקוח לקבלת התשובה מהלקוח והבדיקה של השרת האם תגובת הלקוח נכונה הוא תהליך שלוקח זמן.
   2. יצירת ה CAPTCHA-
      1. יתרונות-
         1. יצירת צירוף אותיות ומספרים אקראי כל פעם מחדש, כמו כן קשקוש אקראי על התמונה.
      2. חסרונות-
         1. יצירת צירוף אותיות בעל משמעות גסה. דרך פתרון: שמירת מאגר מילים גסות ובדיקה כי אף אחת מהמילים לא מופיעה.
4. זיהוי הבקשות של הלקוח ותגובות השרת
   1. יצירת הודעה בתבנית מוסכמת על ידי הלקוח-
      1. יתרונות-
         1. ייעול הקוד
      2. חסרונות-
         1. שימוש בתבנית קבועה מגביל את המידע שניתן להעביר בהודעה
   2. זיהוי הפקודות על ידי השרת-
      1. יתרונות-
         1. הבנה טובה של השרת והלקוח.
      2. חסרונות-
         1. לא מוגן מפני צד שלישי. מאוד קל לזהות מהן הפקודות המוסכמות.
5. שליחת דואר אלקטרוני
   1. שימוש בחשבון Outlook
      1. יתרונות-
         1. ניהול חשבון בצורה נוחה.
      2. חסרונות-
         1. נתון לפרצה.
   2. שימוש בפרוטוקול Smtp
      1. יתרונות-
         1. אוטומציה פשוטה ונוחה לשליחת מייל
         2. תמיכה בפרוטוקול SSL
      2. חסרונות-
         1. פרטי ההתחברות למייל מופיעים בקוד
6. ייעול זמן תגובה של התקשורת
   1. התחברות מוקדמת לבסיס הנתונים
      1. יתרונות-
         1. ייעול זמן התגובה של התקשורת
      2. חסרונות-
         1. הרצת תהליכים לתקופה ללא ידיעה מתי יהיה צורך בשימוש בסיס הנתונים.
   2. שליחה מוקדמת ללקוח
      1. יתרונות-
         1. ייעול זמן התגובה של התקשורת
         2. דילוג על זמן הכנסת הפרטים למערכת
      2. חסרונות-
         1. השרת אינו בטוח כי הכנסת הנתונים הצליחה ולמרות חוסר הוודאות שולח השרת אישור להתחברות הלקוח

**הפתרון שנבחר**

כעת נפרט יותר על הפתרונות שממשנו, למה ממשנו אותם, איך ממשנו אותם בקוד, ונביא דוגמאות.

1. ביצוע תקשורת עם מספר לקוחות בו-זמנית  
תחילה השרת פותח את האזנה ומחכה ללקוחות שיתחברו. ברגע החיבור השרת מייצר עצם של המחלקה Client.

|  |
| --- |
| *// Start listen to incoming connection requests*  listener.Start();  *// infinite loop.*  while (true)  {  *// creating object Client so the server will be able to server multiple client at the same time.*  Client user = new Client(listener.AcceptTcpClient());  } |

לאחר יצירת האובקייט Client מתחיל תהליך התקשורת בין אותו לקוח לשרת.

|  |
| --- |
| class Client  {  public static Hashtable AllClients = new Hashtable();  private TcpClient \_client;  private string \_clientIP;  private byte[] data;  private ClientRSA rsa;  private string captchaString;     public Client(TcpClient client)  {  ...  } |

לכל לקוח השרת מייצר TCPClient משלו ועל ידי כך פותח Socket לכל לקוח.

2. הצפנה - RSA  
בתחילת התקשורת שני הצדדים מעבירים אחד לשני את המפתח הציבורי.

קוד העברת המפתח הציבורי וקבלת המפתח הציבורי של הצד השני:

|  |
| --- |
| *//get server public key parameters*  public static void GetParameters()  {  string ServerXmlString = HandleConnection.GetData();  ServerRsa.FromXmlString(ServerXmlString);  }   *//sends my public key parameters to the server*  public static void SendParameters()  {  string XmlString = ClientRsa.ToXmlString(false);  HandleConnection.SendData(XmlString);  } |

לאחר מכן, כל מידע שעובר בין השרת והלקוח עובר קודם כל הצפנה ולאחר מכן פענוח:

פעולות הצפנה ופענוח:

|  |
| --- |
| *//encrypt data using server public key and return it as a string*  public static string Encrypt(string msg)  {  byte[] data = System.Text.Encoding.ASCII.GetBytes(msg);  data = ServerRsa.Encrypt(data, false);  return Convert.ToBase64String(data);  }   *//decrypt data using client private key and return it as a string*  public static string Decrypt(string msg)  {  byte[] data = Convert.FromBase64String(msg);  data = ClientRsa.Decrypt(data, false);  return System.Text.Encoding.ASCII.GetString(data);  } |
|  |

חלק מההמרות נעשות בפורמט מחרוזת מבסיס 64 על מנת למנוע פגימה במידע המועבר

3. יצירת CAPTCHA - בדיקת אנושיות - בצד שרת והעברתה ללקוח

השרת מייצר את הCAPTCHA מיד לאחר החלפת המפתחות בינו לבין הלקוח

|  |
| --- |
| public Client(TcpClient client)  {  ...  Captcha captcha = new Captcha();   captchaString = captcha.GetString();  captcha.GetBitmap().Save(@"c:\Simple.bmp");  sendImage(captcha.GetBitmap());    } |

המחלקה CAPTCHA:

|  |
| --- |
| class Captcha  {  private static Random rnd = new Random();  private string rndText = string.Empty;  private Bitmap bitmap;   public Captcha()  {  rndText = GetRandomText();  bitmap = CreateImage();  }   public Bitmap CreateImage()  {  *//using all the functions in class to make the image*  ...  }   private static void DrawRandomLines(Graphics g)  {  *//Creating Lines on The Captcha*  ...  }   private static Point[] GetRandomPoints()  {  *//making random points for the position of the lines*  ...  }   public static string GetRandomText()  {  *//making random text*  ...  }  public string GetString()  {  return rndText;*//return random string*  }   public Bitmap GetBitmap()  {  return bitmap;*//return bitmap file*  }  } |

לאחר יצירת התמונה השרת שולח אותה ללקוח. הלקוח מקבל את התמונה ומציג אותה.

|  |
| --- |
| //Client Captcha code  public class Captcha  {  private PictureBox pb;*//the target pictureBox*   public Captcha(PictureBox pb)  {  this.pb = pb;  CreateImage(pb);  }   public void CreateImage(PictureBox pb)  {  Byte[] Byte = HandleConnection.GetRawData();  MemoryStream memstr = new MemoryStream(Byte);  Image img = Image.FromStream(memstr);   pb.Size = img.Size;   pb.Image = img;  }   public bool CheckAnswer(String Answer)  {  HandleConnection.SendEncryptedData("CAPTCHA;" + Answer);  if (HandleConnection.GetEncryptedData() == "YES")  return true;  return false;  }   public void Terminate()  {  pb.Image = null;  }  } |

כאשר הלקוח שולח לשרת את תשובתו, השרת מאמת אותה ושולח לו האם היא נכונה או לא.

|  |
| --- |
| public void Captcha(string[] str)  {  string captchaString = client.GetCaptchaString();  if (captchaString == str[1])  client.SendMessage("YES");  else  client.SendMessage("NO");  } |

4.זיהוי הבקשות של הלקוח ותגובות השרת-

כאשר השרת מקבל הודעה מהלקוח הוא מעביר אותה למחלקת הניתוח של ההודעה שם הוא עובר קודם כל פיענוח ולאחר מכן עובר מספר שאילתות בכדי לדעת לאיפה הוא מיועד.

|  |
| --- |
| class AnalyzeMessage  {  private Client client;  private string[] substring;    public AnalyzeMessage(string Message, Client client)  {  this.client = client;   if (Message.Contains("<RSAKeyValue>"))  {  this.client.GetRsa().SetClientParameters(Message);  *//set client rsa parameters*  }  else  {  Message = this.client.GetRsa().Decrypt(Message);  *//decrypt client message to analyze the message*  if (!Message.Contains("OR") && !Message.Contains("'"))  *//secure from SQL injection*  {  char delimiter = ';';  substring = Message.Split(delimiter);  switch (substring[0])*//analyzing the message*  {  case "REGISTER":  Register(substring);  break;   case "LOGIN":  Login(substring);  break;   case "UPDATE":  SQLDB.Update(substring);  break;   case "SELECT":  Select(substring);  break;   case "DELETE":  SQLDB.Delete(substring);  break;   case "VERIFICATION":  Verification(substring);  break;   case "NEWCODE":  NewCode(substring);  break;   case "CAPTCHA":  Captcha(substring);  break;   default:  break;  }  }  else  client.SendMessage("false");  }   }  } |

5.שליחת דואר אלקטרוני-

לצורך שליחת הדואר האלקטרוני ישנן שתי מחלקות, האחת יוצרת את תוכן ההודעה ומוסיפה את הקוד שעל הלקוח לכתוב אל הבסיס נתונים.

|  |
| --- |
| class EmailVerification  {  private readonly string email;  private string code;  private string[] str;   public EmailVerification(string email, string[] str)  {  this.email = email;  this.str = str;  SendEmail();  }   private void SendEmail()  {  string mailUser = "SecureATM@outlook.com";  string mailUserPwd = "aPOy4862%%";   var sender = new OutlookMail(mailUser, mailUserPwd);  sender.SendMail(email, "Test Mail", "Hello! \n Your code is: " + GetRandomText());   }    public string GetRandomText()  {  …  //making random text and insert the text in to the database  }  } |

והמחלקה השניה מייצרת את ההתחברות אל חשבון הoutlook ושולחת את הודעה אל המייל הדרוש.

|  |
| --- |
| class OutlookMail  {  string \_sender="";  string \_password="";  public OutlookMail(string sender, string password)  {  \_sender = sender;  \_password = password;  }   public void SendMail(string recipient, string subject, string message)  {  SmtpClient client = new SmtpClient("smtp-mail.outlook.com");   client.Port = 587;  client.DeliveryMethod = SmtpDeliveryMethod.Network;  client.UseDefaultCredentials = false;  System.Net.NetworkCredential credentials =  new System.Net.NetworkCredential(\_sender, \_password);  client.EnableSsl = true;  client.Credentials = credentials;   try  {  var mail = new MailMessage(\_sender.Trim(), recipient.Trim());  mail.Subject = subject;  mail.Body = message;  client.Send(mail);  }  catch (Exception ex)  {  Console.WriteLine(ex.Message);  }  }  } |

6.ייעול זמן תגובה של התקשורת-

בכדי לייעל את מירב הזמן השרת מבצע התחברות אל בסיס הנתונים כבר בתחילת ריצת הפרויקט.

|  |
| --- |
| *// Start listen to incoming connection requests*  listener.Start();  *// open the connection between the server and the database*  SQLDB.Start(); |

**דרישות ומגבלות המערכת**

דרישות

על מנת שהתכנית תפעל ישנן כמה דרישות שצריכות להיענות קודם לכן:

* מערכת הפעלה Windows XP ומעלה על המחשב.
* Visual Studio על המערכת.

מוגבלויות

* תמיכה בשפה האנגלית בלבד (שימוש בASCII ולא UTF).
* הנחה בסיסית שהלקוח יודע את הIP והפורט שעליהם מאזין השרת.
* הגבלה של 256 בייטים בסטרים בין השרת ללקוח.
* הגבלה לסכום כסף לפי ערך המקסימום של משתנה מסוג float.
* יכול להיווצר דיליי בתקשורת בין הלקוח לשרת בתחילת התקשורת עקב החלפת המפתחות הציבוריים.

**ממשק משתמש**



**עץ מודולים**



\*תקשורת עם הלקוח נעשית במודול Client.cs



\*תקשורת עם השרת נעשית ממודול HandleConnection.cs

צד שרת-

Client.cs - המחלקה מייצרת את התקשורת בין השרת ללקוח. המחלקה גם אחראית על מעבר כל ההודעות מהלקוח לשרת וההיפך. תחילה המחלקה מקבלת ושולחת את מפתחות ההצפנה ולאחר מכן מייצרת את הCAPTCHA ושולחת אותו גם כן ללקוח.

ClientRSA.cs - המחלקה שומרת את המפתחות של השרת ומקבלת את מפתחות הלקוח. במחלקה שתי פעולות עיקריות שהן פיענוח והצפנה של מידע.

Captcha.cs - המחלקה מייצרת תמונה ובה רצף אותיות וקווים המיוצרים באופן רנדומלי. המחלקה שומרת את רצף האותיות בתור מחרוזת בכדי לאמת האם הלקוח צדק או לא.

AnalyzeMessage.cs - המחלקה מקבלת את כל הודעות השרת ומחליטה מה לעשות איתן ולאן הן מיועדות. המחלקה מבצעת זאת בעזרת מספר שאילתות בהתאם לסוג ההודעה, המחלקה מתפקדת כנתב ולכן מחוברת למספר רב של מחלקות.

SQLDB.cs - המחלקה מקושרת אל בסיס הנתונים והיא היחידה האחראית לשאוב ולהכניס נתונים, כל שאר המחלקות לא רשאיות לכך. המחלקה היא סטטית כיוון שניתן לגשת ולבקש נתון מכל מחלקה שהיא.

EmailVerification.cs - המחלקה מייצרת רצף אותיות רנדומלי שישלח בעתיד לכתובת דואר האלקטרוני של המבקש, מייצרת אובייקט OutlookMail ומבקשת מSQLDB להכניס את רצף האותיות לתוך בסיס הנתונים.

OutlookMail.cs - המחלקה שולחת דואר אלקטרוני לכתובת דואר האלקטרוני של המבקש. המחלקה מנסחת את ההודעה ומוסיפה את רצף האותיות. המחלקה מקבלת את שם המשתמש והסיסמה של השולח ושולחת לנמען.

צד לקוח-

Captcha.cs - המחלקה אחראית על בקשת התמונה של בדיקת האנושיות מהשרת וקבלת התמונה. מיוצר אובייקט Captcha שבהתאם לפלט המשתמש ניתן לבדוק האם הוא בן אנוש או מכונה באופן די מדויק.

Login.cs - המחלקה אחראית על התחברות הלקוח למשתמש שלו.  
אם ללקוח אין משתמש קיים המחלקה מפנה אותו למודול ה Register.cs.  
המחלקה שולחת את הקלט שהמשתמש מכניס לה לשרת ורק אם השרת מאשר את התחברותו הלקוח מקבל גישה למשתמש שלו.

Register - המחלקה אחראית על תהליך ההרשמה של משתמש חדש.  
על המשתמש להכניס שם משתמש, ססמה, ססמה חוזרת לווידוא וכתובת דואר אלקטרוני אשר עומדים בקריטריונים להרשמה תקנית. אם התהליך צלח המחלקה יוצרת אובייקט חדש של המחלקה EmailVerification.

EmailVerification.cs - המחלקה אחראית על תהליך אישור כתובת הדואר האלקטרוני. נשלח דואר אלקטרוני לכתובת שנרשמה דרך השרת, והמחלקה שולחת לשרת את הקלט שהמשתמש מכניס לה (הקוד הנשלח במייל). במידה והקוד נכון המחלקה מחזירה את המשתמש למחלקת Login.cs.

Main.cs - המחלקה שמציגה למשתמש את הדף הראשי של התוכנה.  
המשתמש יכול להכניס ולמשוך "כסף" אל החשבון ומחוצה לו. כל פעולה של המשתמש נשלחת לשרת ולאחר כל פעולה סכום הכסף של המשתמש מעודכן לפי השרת.

Constants.cs - מחלקה סטטית המכילה משתנים ידועים כגון כתובות, פורטים וכו'.

RSA.cs - מחלקה סטטית המכילה פעולות ומשתנים הקשורים להצפנת הRSA. המחלקה יכולה לשלוח את מפתח הRSA הציבורי של הלקוח לשרת ולקבל את המפתח הציבורי של השרת, כמו כן להצפין הודעות באמצעות המפתח הציבורי של השרת ולפענח הודעות מוצפנות באמצעות המפתח בפרטי של הלקוח.

HandleConnection.cs - מחלקה סטטית אשר אחראית על התקשורת עם השרת. אחראית להאזנה לשרת ולשליחת מידע. עובדת בשימוש קבוע במחלקת RSA.cs להצפנת המידע המועבר.

**תיעוד קוד**

צד שרת

Client.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| Public Client(TcpClient client) | הפעולה מקבלת אובייקט מטיפוס TcpClient | - | הפעולה הבונה של המחלקה |
| public void SendMessage(string message) | הפעולה מקבלת מחרוזת המיועדת לשליחה ללקוח | - | הפעולה מצפינה ושולחת את ההודעה אל הלקוח |
| public void ReceiveMessage(IAsyncResult ar) | הפעולה מקבלת אובייקט מטיפוס IAsyncResult | - | הפעולה מאזינה ללקוח וברגע קבלת ההודעה היא ממירה אותה למחרוזת ומעבירה אותה אל המחלקה AnalyzMessage |
| public void sendImage(Bitmap b) | הפעולה מקבלת תמונה מטיפוס bitmap | - | הפעולה שולחת את הCAPTCHA ללקוח בתור תמונה |
| public ClientRSA GetRsa() | - | הפעולה מחזירה אובייקט של מחלקה ClientRSA | הפעולה מעבירה את ClientRSA למחלקות שלא רשאיות לגשת אליה |
| public string GetCaptchaString() | - | הפעולה מחזירה מחרוזת | הפעולה מחזירה מחרוזת לצורך בדיקת תשובת הלקוח לCAPTCHA |

ClientRSA.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public ClientRSA() | - | - | הפעולה הבונה של המחלקה |
| public string GetServerParameters() | - | הפעולה מחזירה מחרוזת | הפעולה מחזירה את פרמטרים ההצפנה של השרת בכדי לשלוח ללקוח |
| public void SetClientParameters(string clientParameters) | הפעולה מקבלת מחרוזת | - | הפעולה מקבלת את פרמטרים ההצפנה של הלקוח |
| public string Encrypt(string msg) | הפעולה מקבלת מחרוזת | הפעולה מחזירה מחרוזת | הפעולה מצפינה את המחרוזת שהיא מקבלת ומחזירה את המחרוזת המוצפנת |
| public string Decrypt(string msg) | הפעולה מקבלת מחרוזת | הפעולה מחזירה מחרוזת | הפעולה מפענחת את המחרוזת שהיא מקבלת ומחזירה את המחרוזת המפוענחת |

Captcha.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public Captcha() | - | - | הפעולה הבונה של המחלקה |
| public Bitmap CreateImage() | - | הפעולה מחזירה תמונה מטיפוס Bitmap | הפעולה מייצרת את הCAPTCHA בעזרת פעולות המחלקה |
| private static void DrawRandomLines(Graphics g) | הפעולה מקבלת אובייקט מטיפוס graphics | - | הפעולה מציירת קווים רנדומלים על התמונה שמקבלת |
| private static Point[] GetRandomPoints() | - | הפעולה מחזירה מערך של נקודות | הפעולה קובעת נקודות רנדומליות בכדי למקם את הקווים |
| public static string GetRandomText() | - | הפעולה מחזירה מחרוזת | הפעולה מייצרת מחרוזת רנדומלית לצורך הכיתוב על התמונה |
| public string GetString() | - | הפעולה מחזירה מחרוזת | הפעולה מחזירה את המחרוזת שנקבעה בתור הכיתוב על התמונה |
| public Bitmap GetBitmap() | - | הפעולה מחזירה תמונה מטיפוס Bitmap | הפעולה מחזירה את תמונת הCAPTCHA בשלמותה |

AnalyzeMessage.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public AnalyzeMessage(string Message, Client client) | הפעולה מקבלת מחרוזת ואובייקט מטיפוס Client | - | הפעולה הבונה של המחלקה |
| public void Register(string[] str) | הפעולה מקבלת מערך של מחרוזות | - | הפעולה מקושרת אל המחלקהSQLDB |
| public void Login(string[] str) | הפעולה מקבלת מערך של מחרוזות | - | הפעולה מקושרת אל המחלקהSQLDB |
| public void Select(string[] str) | הפעולה מקבלת מערך של מחרוזות | - | הפעולה מקושרת אל המחלקהSQLDB |
| public void Verification(string[] str) | הפעולה מקבלת מערך של מחרוזות | - | הפעולה מקושרת אל המחלקהSQLDB |
| public void NewCode(string[] str) | הפעולה מקבלת מערך של מחרוזות | - | הפעולה מקושרת אל המחלקהSQLDB |
| public void Captcha(string[] str) | הפעולה מקבלת מערך של מחרוזות | - | הפעולה מקושרת אל המחלקהSQLDB |

SQLDB.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public static void Start() | - | - | הפעולה בודקת האם התבצע חיבור בין השרת לבסיס הנתונים ובמידה ולא הפעולה מבצעת את החיבור |
| public static void Register(string code, string[] str) | הפעולה מקבלת מערך מחרוזות ובנוסף מחרוזת | - | הפעולה מכניסה את המשתמש למערכת על פי הנתונים הרשומים במערך המחרוזות ובמחרוזת |
| public static string CheckPassword(string[] str) | הפעולה מקבלת מערך מחרוזות | הפעולה מחזירה מחרוזת | הפעולה בודקת האם הסיסמה ששלח הלקוח זהה לסיסמה הרשומה בבסיס הנתונים |
| public static void Update(string[] str) | הפעולה מקבלת מערך מחרוזות | - | הפעולה מעדכנת שתי עמודות בבסיס הנתונים במידת הצורך (כסף או מחרוזת הזדהות) |
| public static string Select(string[] str) | הפעולה מקבלת מערך מחרוזות | הפעולה מחזירה מחרוזת | הפעולה מחזירה את יתרת הלקוח כמחרוזת |
| public static void Delete(string[] str) | הפעולה מקבלת מערך מחרוזות | - | הפעולה מוחקת את המשתמש מבסיס הנתונים |
| public static bool Verification(string[] str) | הפעולה מקבלת מערך מחרוזות | הפעולה מחזירה משתנה בוליאני | הפעולה בודקת האם האיבר השלישי במערך המחרוזות זהה למחרוזת ההזדהות בבסיס הנתונים |
| public static void UpdateVerifitaion(string[] str) | הפעולה מקבלת מערך מחרוזות | - | הפעולה משנה את מחרוזת ההזדהות של המשתמש לארבע כוכביות |
| public static bool CheckIfVerificationNull(string[] str) | הפעולה מקבלת מערך מחרוזות | הפעולה מחזירה משתנה בוליאני | הפעולה בודקת האם הלקוח הזדהה באמצעות מחרוזת ההזדהות |
| public static bool UserNameDoesNotExist(string[] str) | הפעולה מקבלת מערך מחרוזות | הפעולה מחזירה משתנה בוליאני | הפעולה בודקת האם שם המשתמש קיים במערכת |

EmailVerification.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public EmailVerification(string email, string[] str) | הפעולה מקבלת מערך מחרוזות ובנוסף מחרוזת | - | הפעולה הבונה של המחלקה |
| private void SendEmail() | - | - | הפעולה מייצרת אובייקט של מחלקה OutlookMail ומנסחאת תוכן הודעה |
| public string GetRandomText() | - | הפעולה מחזירה מחרוזת | הפעולה מייצרת מחרוזת רנדומלית ומכניסה אותה לבסיס הנתונים |

OutlookMail.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public OutlookMail(string sender, string password) | הפעולה מקבלת 2 מחרוזות | - | הפעולה הבונה של המחלקה |
| public void SendMail(string recipient, string subject, string message) | הפעולה מקבלת 2 מחרוזות | - | הפעולה מייצרת הודעת דואר אלקטרוני ושולחת אותו לכתובת הדואר האלקטרוני של הנמען |

**צד לקוח**

Captcha.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public Captcha(PictureBox pb) | מקבלת את הpictureBox אשר יכיל את ה Captcha | פעולה בונה | מאתחלת את המחלקה וקוראת לפעולות הנחוצות לבניית ה Captcha. |
| public void CreateImage(PictureBox pb) | מקבלת את הpictureBox אשר יכיל את ה Captcha | - | בונה את תמונת ה captcha ומשתילה אותה ב pictureBox |
| public bool CheckAnswer(String Answer) | מקבלת את התשובה של המשתמש לcaptcha | מחזירה משתנה בוליאני בהתאם לנכונות התשובה | בודקת את תשובת המשתמש |
| public void Terminate() | - | - | סוגרת את המחלקה כראוי |

Login.cs (form)

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public Login(Main main) | חלון Main המקושר להתחברות הנכחית | פעולה בונה | מאתחלת את המשתנים ואת הטופס וקוראת לבניית Captcha |
| private void Login\_Load(object sender, EventArgs e) | - | - | מאתחלת את ה Captcha |
| private void Login\_FormClosed(object sender, FormClosedEventArgs e) | - | - | קוראת לפעולה Terminate |
| private void CaptchaBTN\_Click(object sender, EventArgs e) | - | - | אם הCaptcha נכונה:  מאתחלת את חלון הLogin.  סוגרת את הCaptcha.  אחרת מראה message box: try again |
| private void LoginBTN\_Click(object sender, EventArgs e) | - | - | מעבירה לשרת את נתוני ההתחברות.  מציגה את חלון ווידוא המייל, הודעה של שגיאת ססמה, או מעבירה את המשתמש לחלון הראשי בהתאם לתגובת השרת. |
| private void RegisterBTN\_Click(object sender, EventArgs e) | - | - | מעבירה את המשתמש לחלון ההרשמה. |

EmailVerification.cs (form)

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public EmailVerification(string Username, string Password) | מקבלת שם משתמש וססמה של המשתמש | פעולה בונה | מאתחלת את המשתנים |
| private void SendBTN\_Click(object sender, EventArgs e) | - | - | שולחת את הקוד שהכניס הלקוח לשרת |
| private void NewCode\_Click(object sender, EventArgs e) | - | - | מבקשת מהשרת קוד חדש |

Register.cs

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public Register() | - | פעולה בונה | מאתחלת את המחלקה והמשתנים |
| private void registerBTN\_Click(object sender, EventArgs e) | - | - | בודקת האם הנתונים שהכניס המשתמש עומדים בקריטריונים ומחזירה תגובה בהתאם. אם המשתמש נתן פרטים קבילים שולחת בקשת הרשמה לשרת ומעבירה את המשתמש לטופס אימות הדואר האלקטרוני. |

Main.cs (form)

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| private void Main\_Load(object sender, EventArgs e) | - | - | החלון הראשי מופעל בתחילת התכנית. התפקיד של הפעולה הזאת הוא לקשר אותו לטופס Login ולדאוג שהוא יהיה בלתי נראה עד לקריאה מהLogin. |
| private void Main\_Load(object sender, EventArgs e) | - | - | מעדכנת את סכום הכסף בחשבון המשתמש לפי נתוני השרת |
| private void DepositBTN\_Click(object sender, EventArgs e) | - | - | מוסיפה סכום כסף שננקב על ידי המשתמש לחשבון על ידי שליחת מידע לשרת |
| private void WithdrawBTN\_Click(object sender, EventArgs e) | - | - | מורידה סכום כסף שננקב על ידי המשתמש לחשבון על ידי שליחת מידע לשרת |
| private void logOutBTN\_Click(object sender, EventArgs e) | - | - | סוגרת את הטופס |

RSA.cs (static)

|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public static void GetParameters() | - | - | מקבלת את המפתח הציבורי מהשרת |
| public static void SendParameters() | - | - | שולחת את המפתח הציבורי של הלקוח לשרת |
| public static string Encrypt(string msg) | מחרוזת המכילה הודעה | מחרוזת המכילה הודעה מוצפנת | מצפינה את המחרוזת המתבקשת לפי המפתח הציבורי של השרת |
| public static string Decrypt(string msg) | מחרוזת המכילה הודעת טקסט מוצפנת | מחרוזת של הטקסט המפוענח | מפענחת מחרוזת טקסט מוצפנת המגיעה מהשרת לפי המפתח הפרטי של הלקוח |

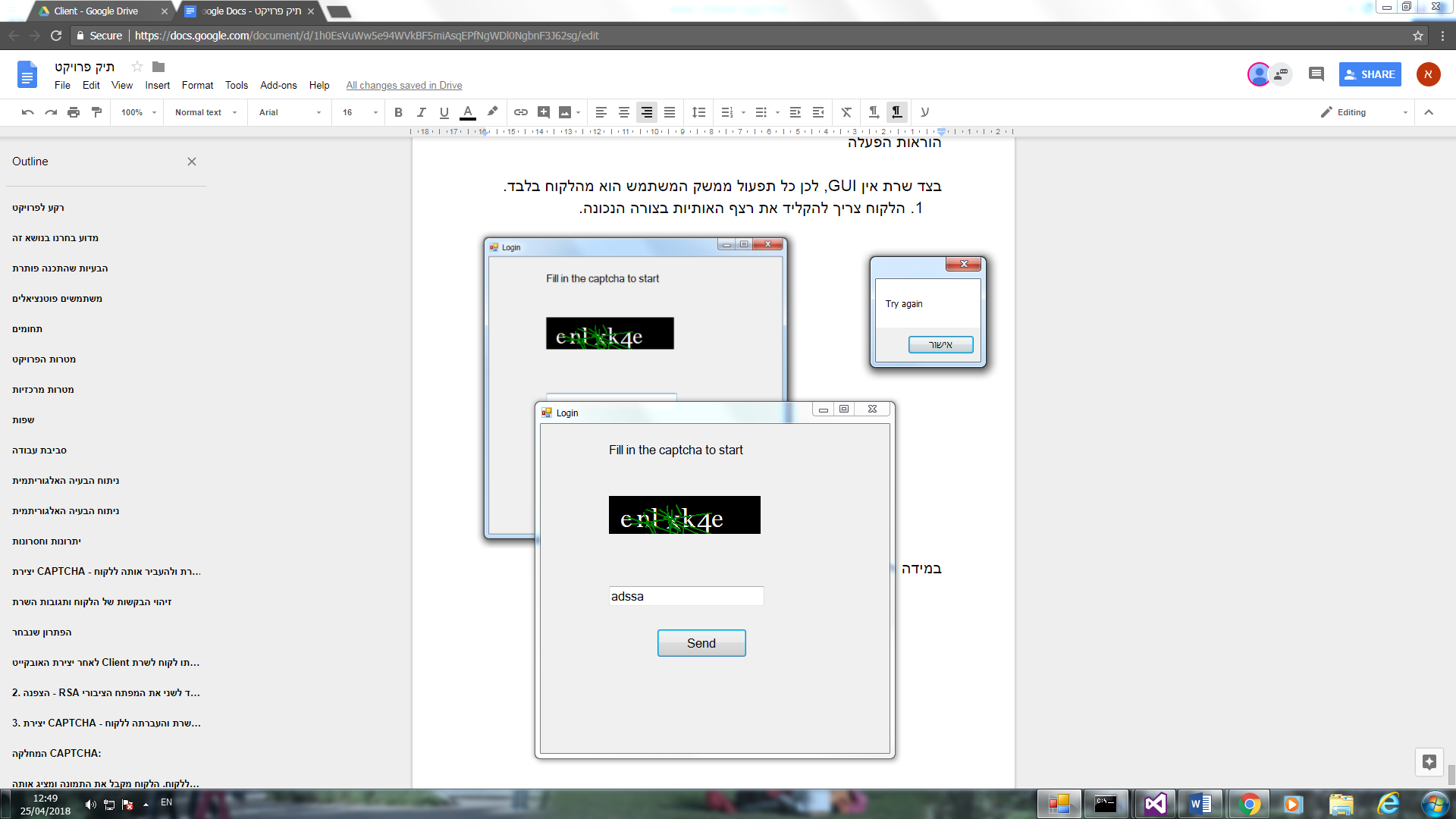
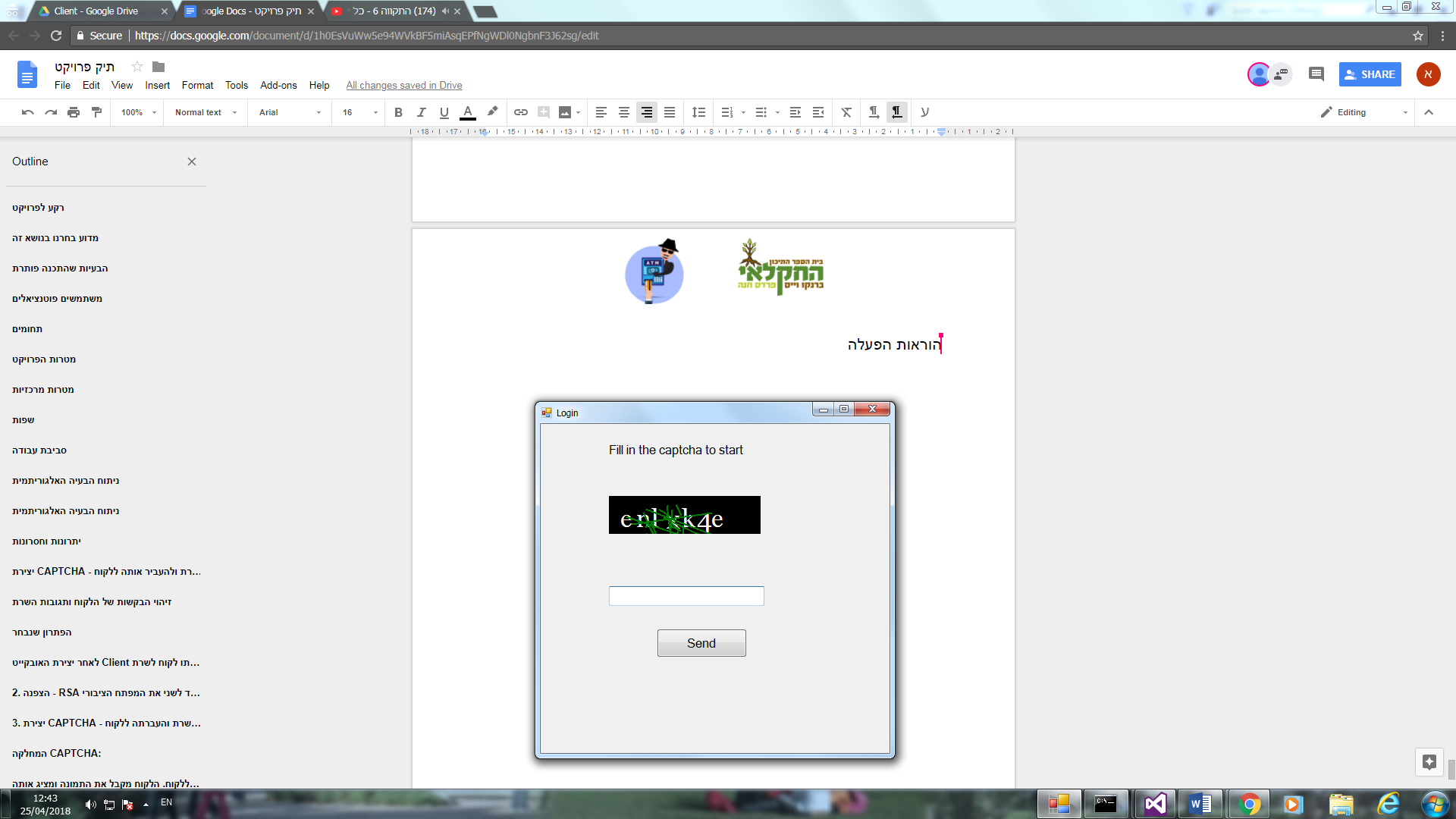
HandleConnection.cs (static)

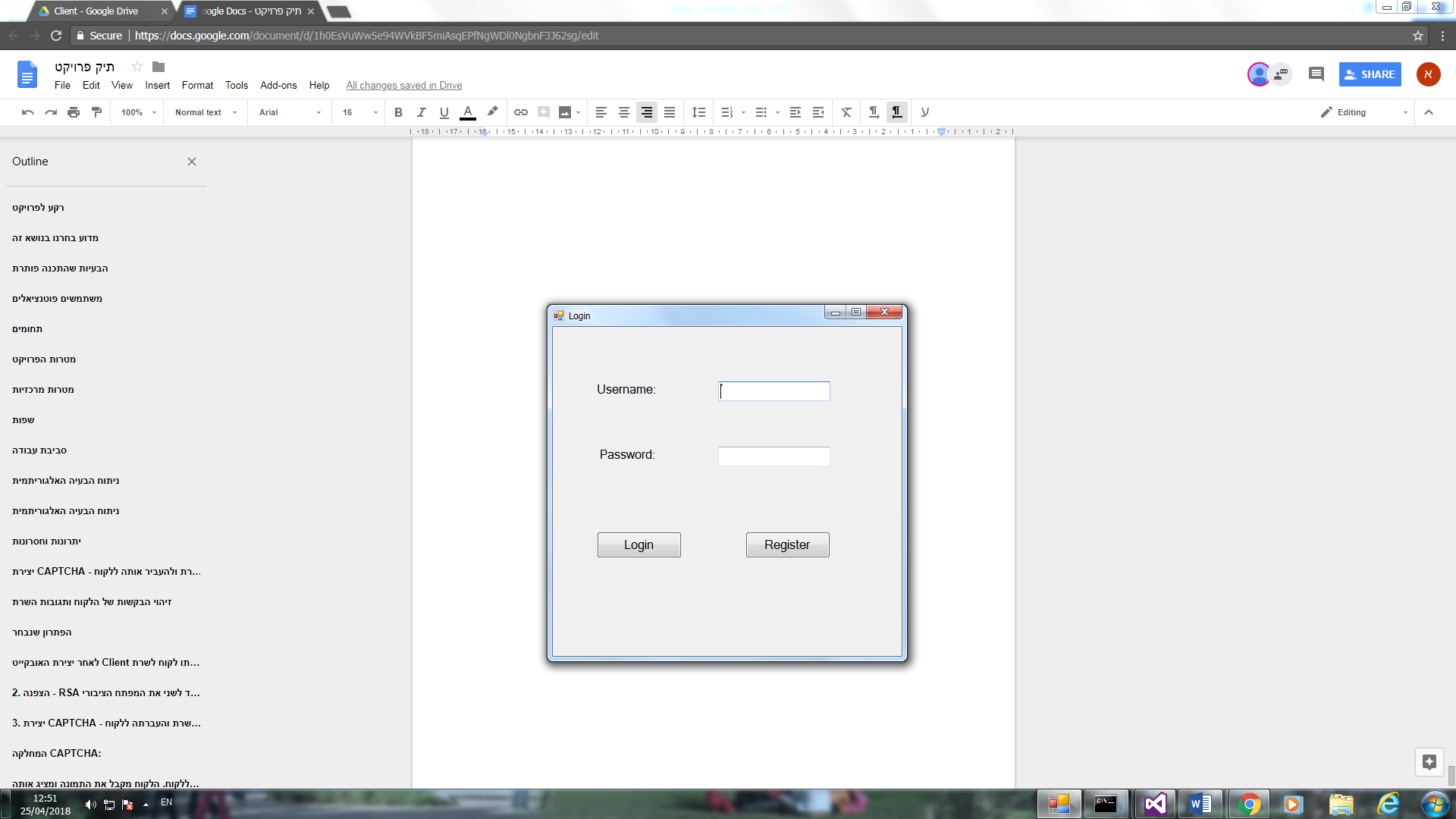
|  |  |  |  |
| --- | --- | --- | --- |
| כותרת | טענת כניסה | טענת יציאה | מה הפעולה עושה? |
| public static void Listen() | - | - | מאזינה לסטרים מהשרת |
| public static void SendData(String data) | מחרוזת המכילה הודעה לשליחה לשרת | - | שולחת הודעה לא מוצפנת לשרת |
| public static void SendEncryptedData(String data) | מחרוזת המכילה הודעה לשליחה לשרת | - | מצפינה הודעה באמצעות מחלקת RSA ושולחת אותה לשרת. |
| public static string GetData() | - | מחרוזת המכילה הודעה לא מוצפנת מהשרת | מקבלת מהשרת מחרוזת המכילה הודעה לא מוצפנת |
| public static string GetEncryptedData() | - | מחרוזת המכילה הודעה מוצפנת מהשרת | מקבלת מהשרת מחרוזת המכילה הודעה מוצפנת ומפענחת אותה בעזרת מחלקת RSA |
| public static void SendRawData(byte[] data) | מערך בייטים לשליחה לשרת | - | שולחת לשרת מערך בייטים לא מוצפן |
| public static byte[] GetRawData() | - | מערך בייטים המגיע מהשרת | מקבלת מערך בייטים לא מוצפן מהשרת |
| public static void Terminate() | - | - | סוגרת את המחלקה כראוי |

**הוראות הפעלה**

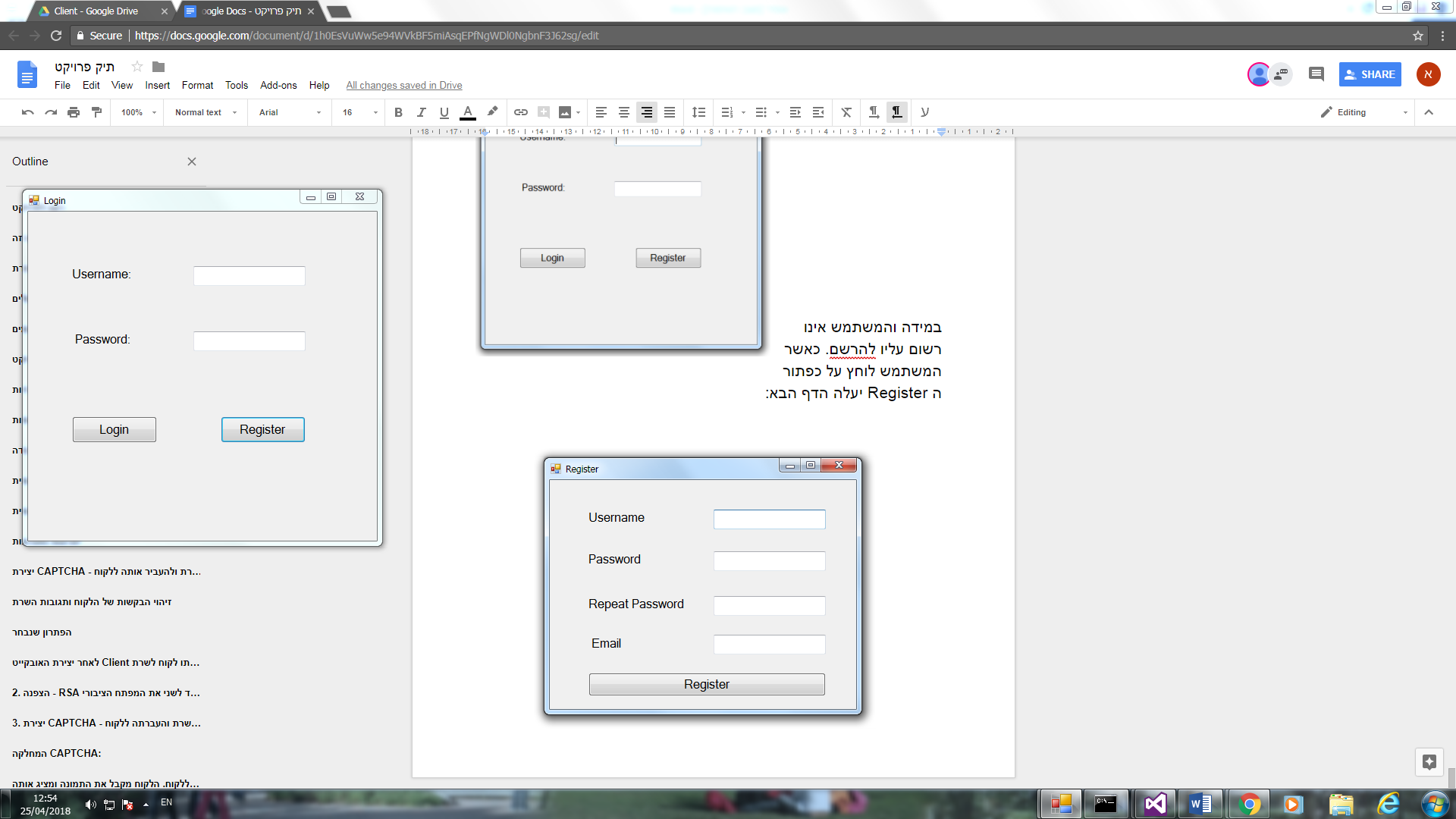
בצד שרת אין GUI, לכן כל תפעול ממשק המשתמש הוא מהלקוח בלבד.

1. הלקוח צריך להקליד את רצף האותיות בצורה הנכונה.

במידה ולא הצליח יופיע החלון:

במידה והצליח הוא עובר אל חלון ההרשמה:

במידה והמשתמש אינו רשום עליו להרשם. כאשר המשתמש לוחץ על כפתור ה Register יעלה הדף הבא:

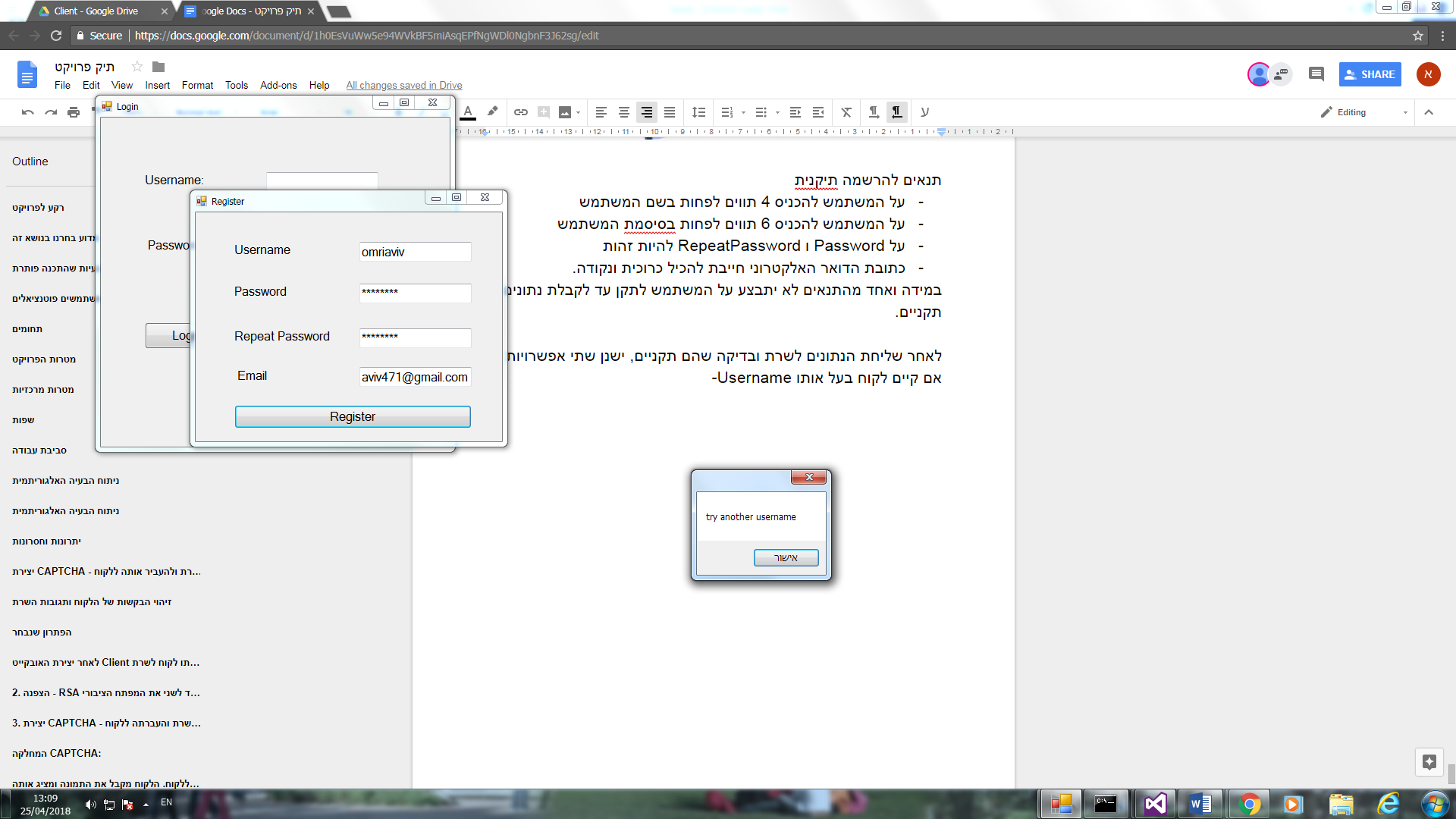


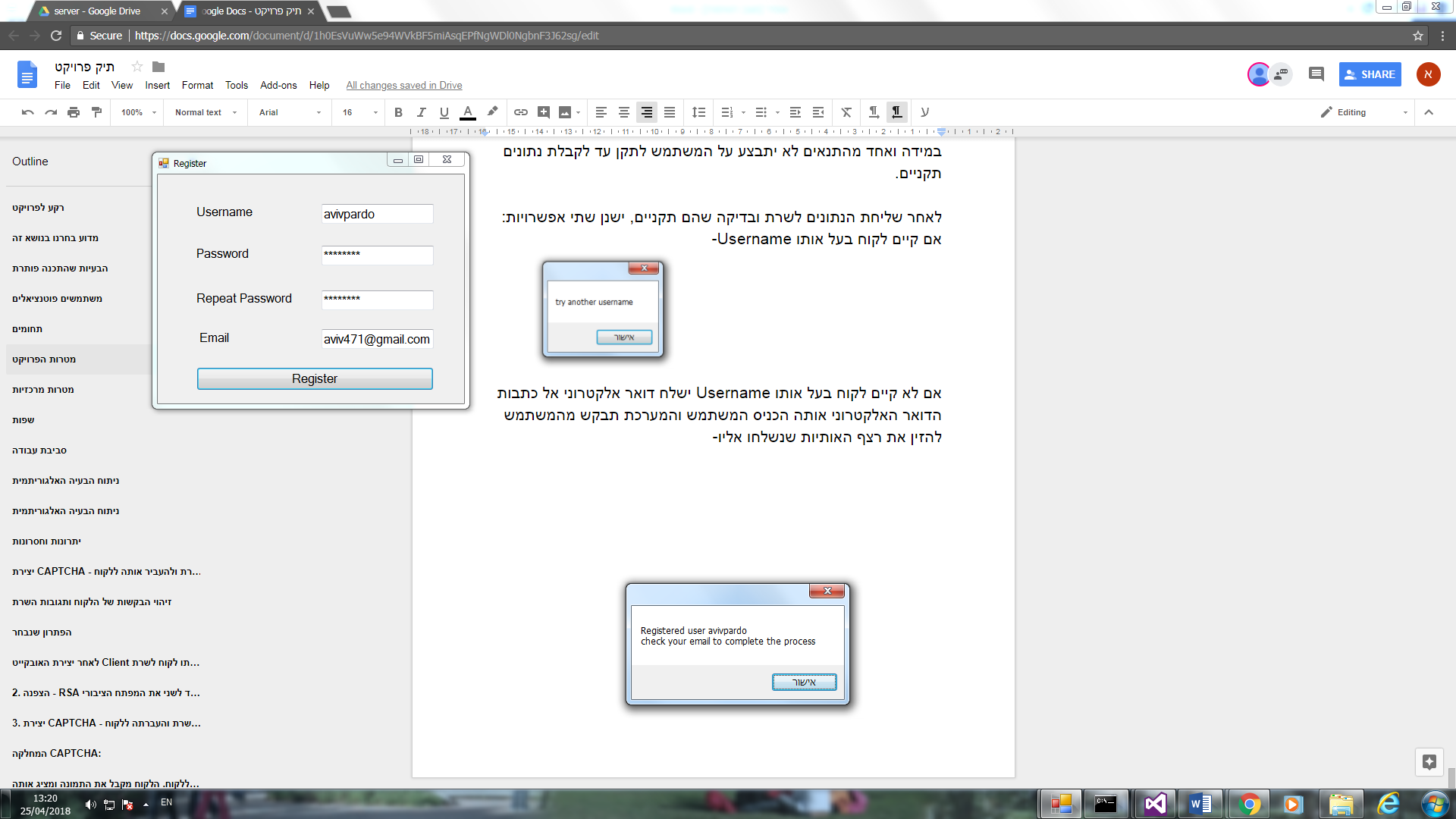
תנאים להרשמה תיקנית

* על המשתמש להכניס 4 תווים לפחות בשם המשתמש
* על המשתמש להכניס 6 תווים לפחות בסיסמת המשתמש
* על Password ו RepeatPassword להיות זהות
* כתובת הדואר האלקטרוני חייבת להכיל כרוכית ונקודה.

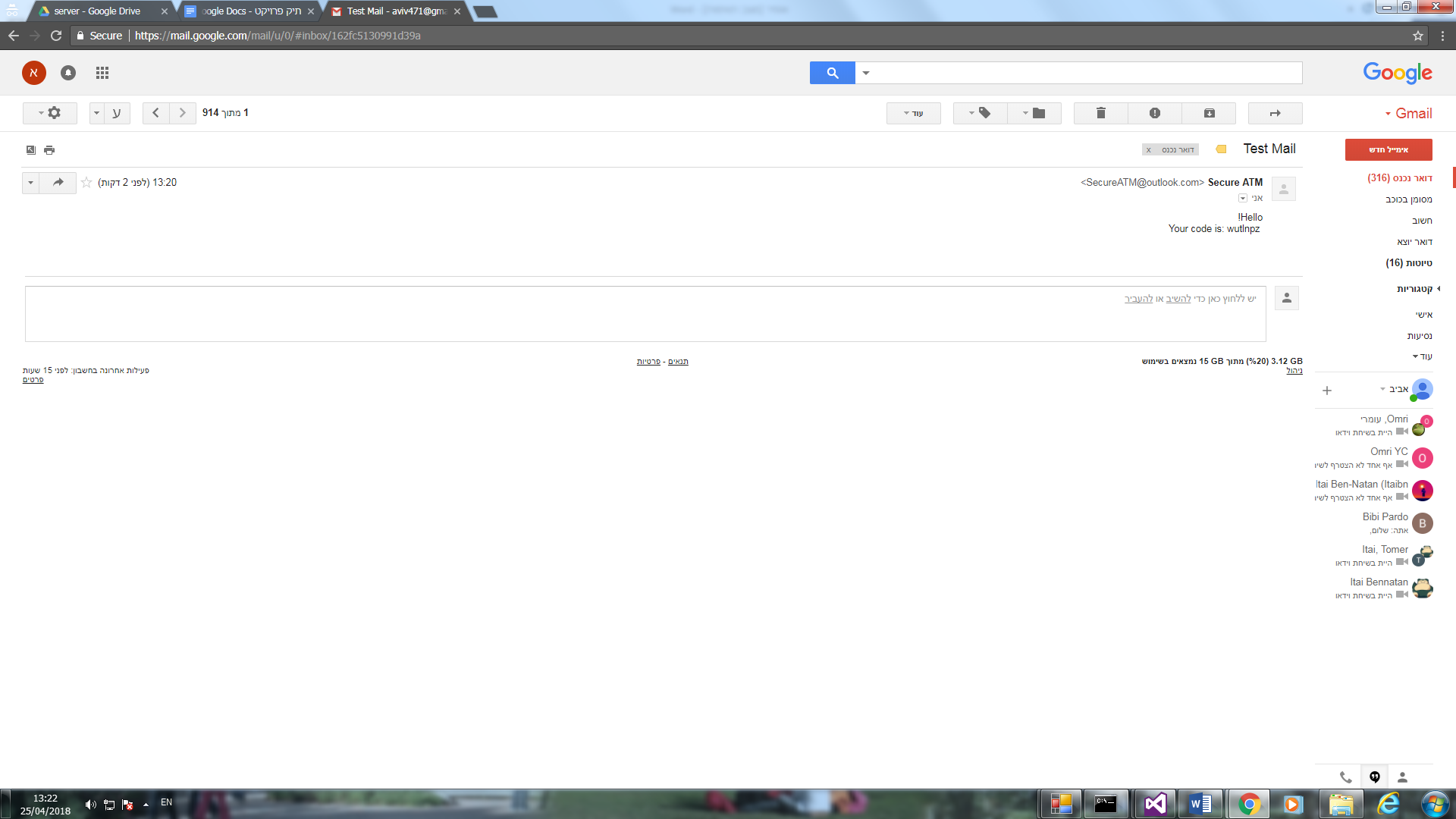
במידה ואחד מהתנאים לא יתבצע על המשתמש לתקן עד לקבלת נתונים תקניים.

לאחר שליחת הנתונים לשרת ובדיקה שהם תקניים, ישנן שתי אפשרויות:

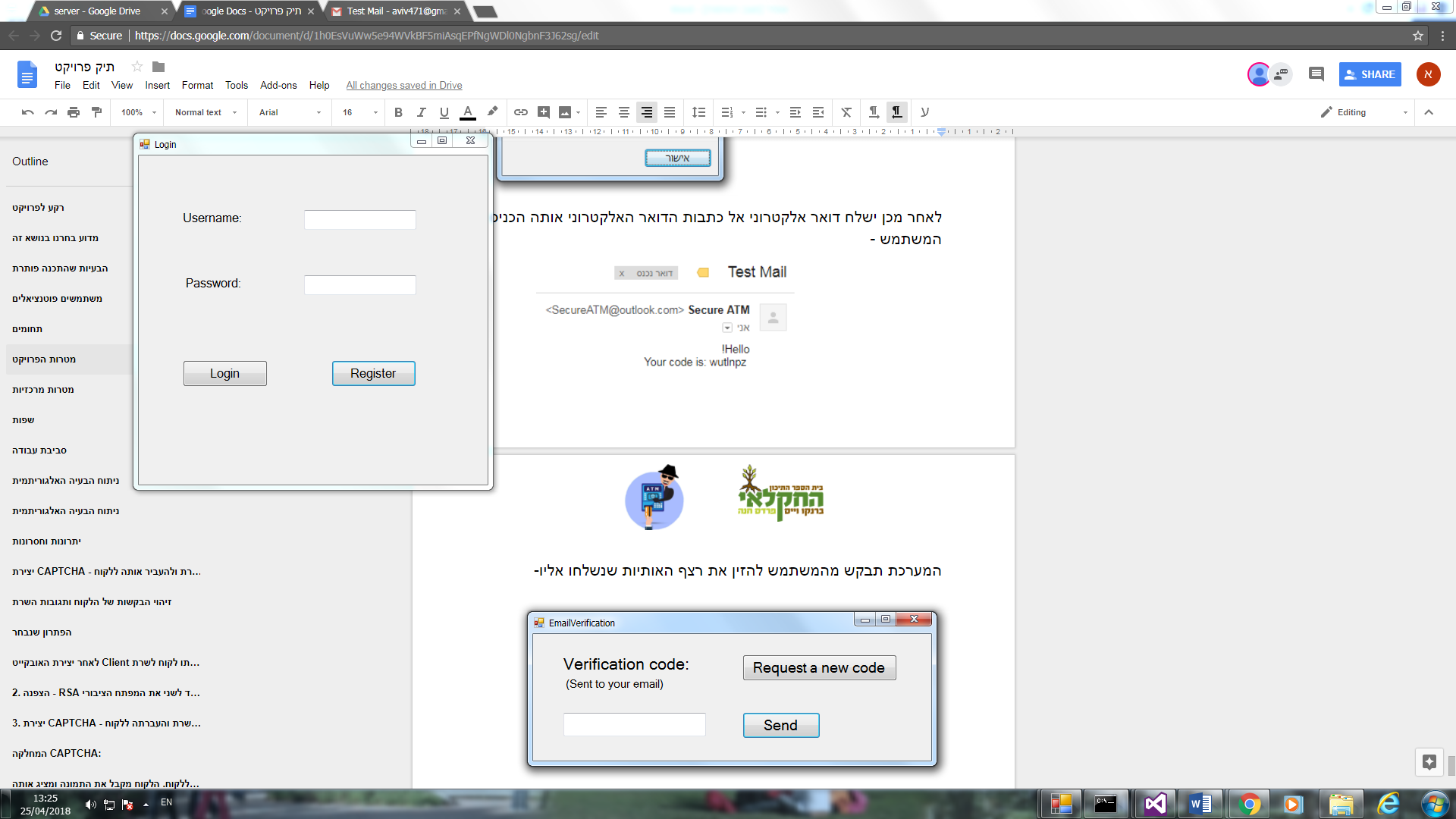
אם קיים לקוח בעל אותו Username- 

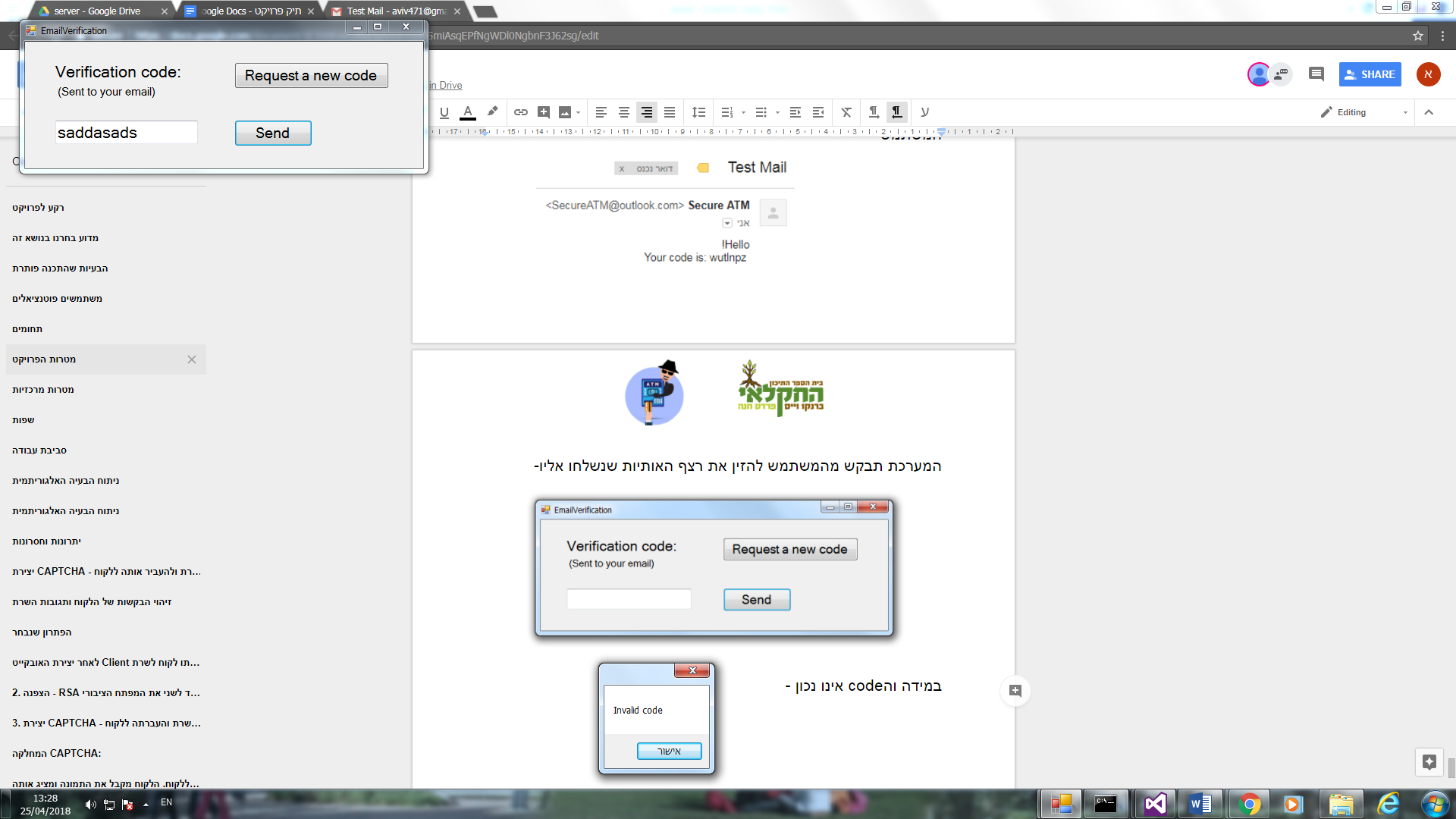


אם לא קיים לקוח בעל אותו Username-

לאחר מכן ישלח דואר אלקטרוני אל כתבות הדואר האלקטרוני אותה הכניס המשתמש - 

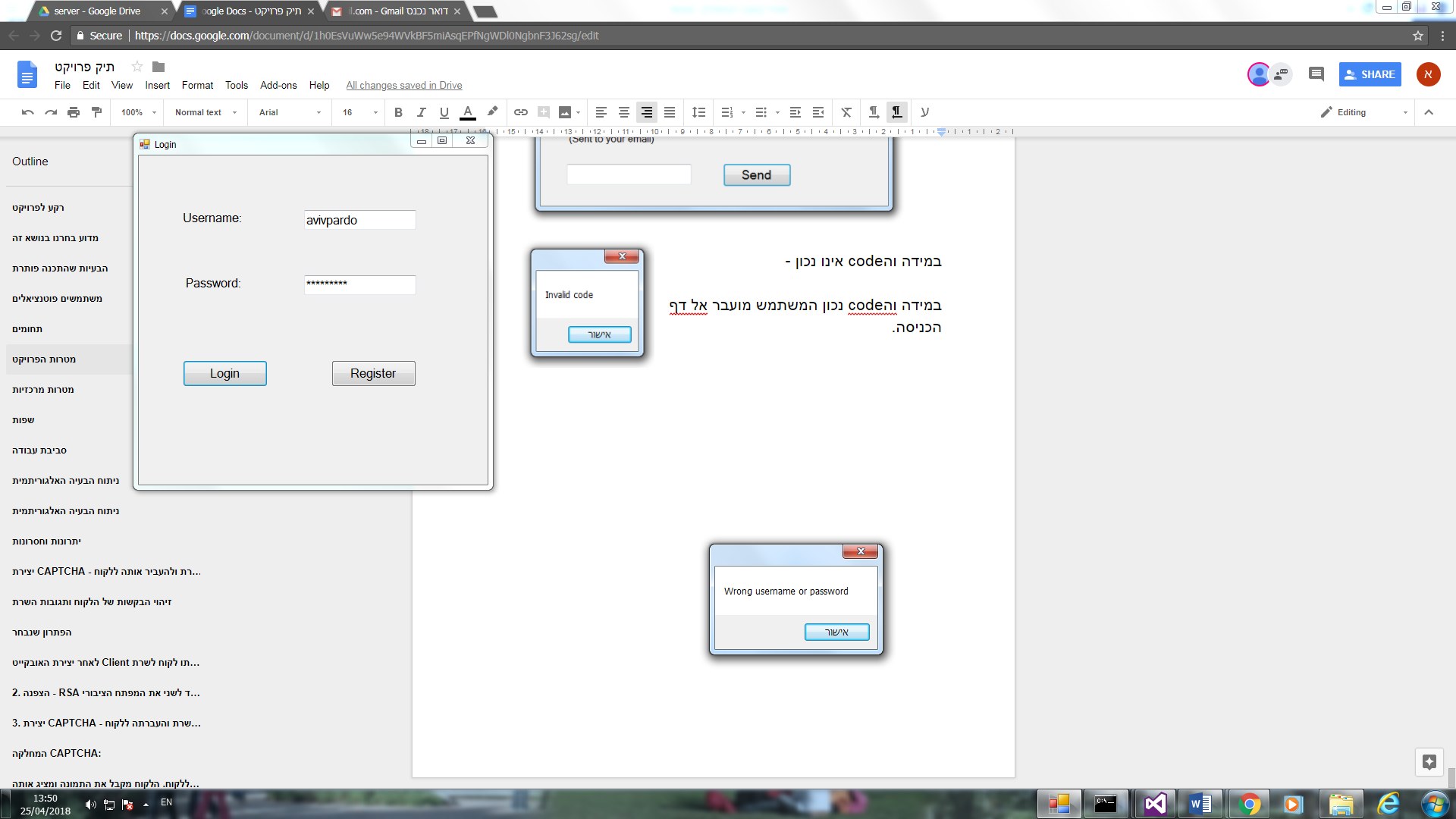
המערכת תבקש מהמשתמש להזין את רצף האותיות שנשלחו אליו-



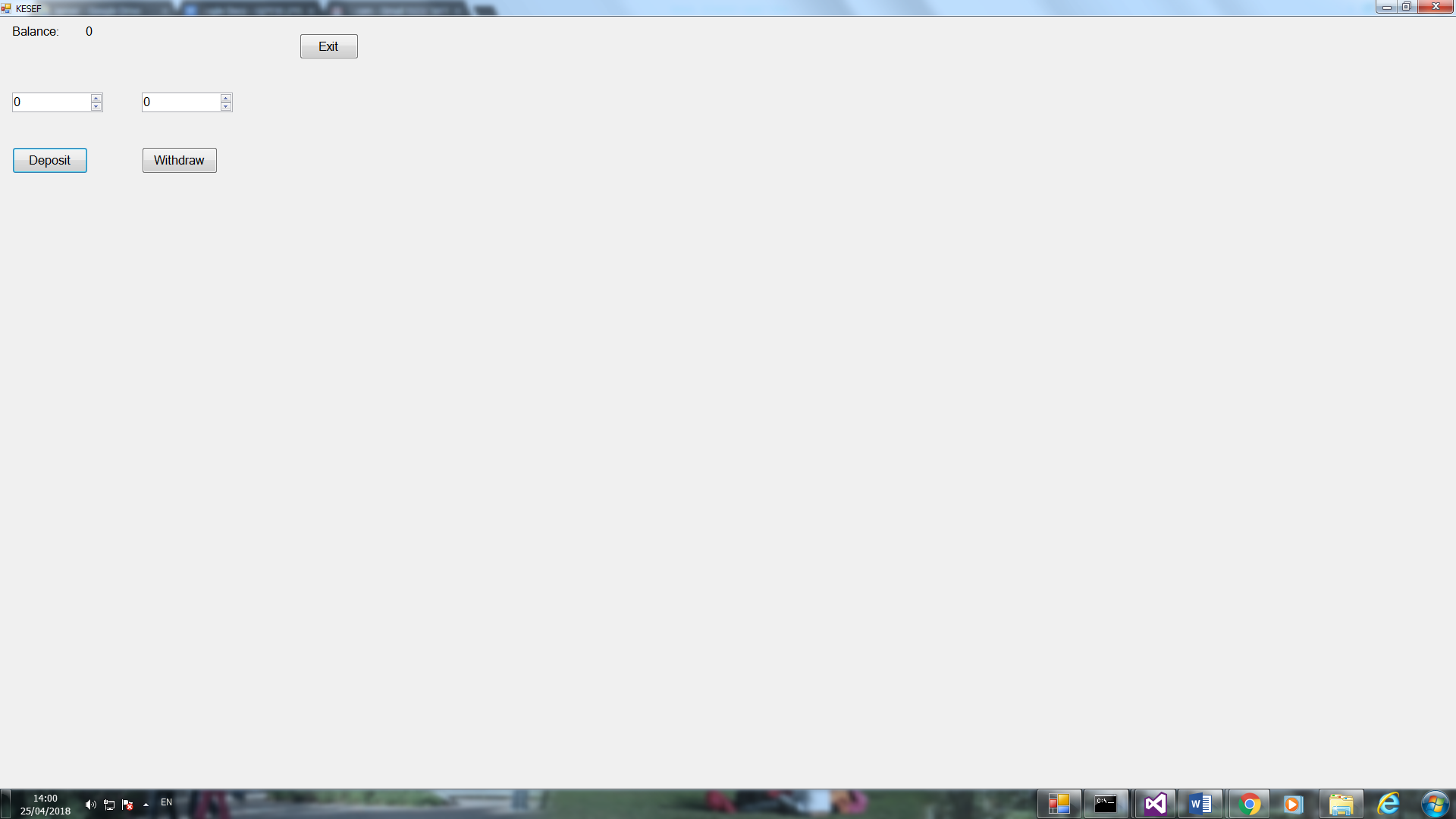


במידה והcode אינו נכון -

במידה והcode נכון המשתמש מועבר אל דף הכניסה.



במידה והלקוח הכניס שם משתמש או סיסמה שגויים עולה החלון-



כאשר המשתמש נכנס לחלון ניהול החשבון יש באפשרותו להוסיף או לפדות כסף.

**סיכום**

למה בחרנו בעבודה זו?

רצינו לממש תקשורת שרת לקוח על פרוטוקול TCP, אך לא ביישום צ'אט רגיל או כל רעיון קונבנציונלי אחר. כמו כן עניין אותנו לעבוד עם בסיס נתונים ועם שפת SQL. הרעיון הזה עלה מתשהו והוא התאים בדיוק למה ששאפנו לעשות.

דברים לשיפור/הוספה

-אובפסקציה בהעברת המפתחות למניעת התחקות של צד שלישי לשרת/לקוח.

-קבצי Log לכל משתמש המתעדים בבירור ופירוט מה נעשה בחשבון למעקב איכותי.

-הוספת סטטיסטיקות לגבי כל חשבון כאופציה בחלון הראשי.

-הוספת פקדונות.

-שיפור ה GUI.

**בבליוגרפיה**

* MSDN
* STACK OVERFLOW
* YOUTUBE
* Code Project