**מרחב עבודה משותף מאובטח בין משתמשים**



**שם המורה:** מיכל כהן

**מגיש:** אופיר איזנברג

**ת"ז:** 315250993

**בית-ספר:** תיכון חדרה

**כיתה:** יב'3

**תאריך:**21.5.17

תוכן עניינים

[רקע לפרויקט 3](#_Toc483097908)

[מטרות הפרויקט 5](#_Toc483097909)

[שפות התכנות וסביבות העבודה 6](#_Toc483097910)

[ניתוח הבעיה האלגוריתמית 7](#_Toc483097911)

[תיאור אלגוריתמים קיימים 9](#_Toc483097912)

[יתרונות וחסרונות 11](#_Toc483097913)

[הפתרון הנבחר 16](#_Toc483097914)

[דרישות ומוגבלויות המערכת 23](#_Toc483097915)

[תיאור ממשק המשתמש והמודולים 24](#_Toc483097916)

[פירוט המודולים בלקוח 26](#_Toc483097917)

[תיאור הממשק למשתמש – הוראות הפעלה 28](#_Toc483097918)

[תיעוד הקוד 36](#_Toc483097919)

[ביבליוגרפיה 62](#_Toc483097920)

[קוד צד השרת 63](#_Toc483097921)

[קוד צד הלקוח 76](#_Toc483097922)

# רקע לפרויקט

הפרויקט שלי הוא מערכת צ'אט מאובטח אשר מאפשרת למשתמשים ברשת המקומית לשוחח אחד עם השני או בקבוצות. התוכנה מאפשרת שיתוף של קבצים ושמירתם בשרת. לכל משתמש רשום יש גישה לעץ קבצים השייך לכל חדר צ'אט שהוא חבר בו. בממשק הקבצים יכול כל משתמש ששייך לצ'אט לשנות את סידור הקבצים בתוך עץ הקבצים על מנת לאפשר גישה נוחה לכל קובץ ששותף בצ'אט בעתיד. ניתן גם לשמור הודעות חשובות בתיקיות משלהן.  
עקרון התוכנה הוא הפיכת הצ'אט שאנו משתמשים בו באופן יום יומי למרחב עבודה הפונה לקהלים שונים.

**בחרתי לעשות את הפרויקט על נושא זה מכמה סיבות:**

1. בתור משתמשת בתוכנות צ'אט אני עדה על בסיס יום יומי להודעות של אנשים שמבקשים שליחה חוזרת של קבצים ששותפו בצ'אט בעבר. לעיתים יש אי נוחות עבור משתמשים אשר נאלצים לשלוח שוב את הקובץ שכבר שלחו ובנוסף עבור משתמשים שנאלצים לקבל פעמיים את אותו הקובץ שתופס להם נפח אחסון במכשיר. רציתי ליצור תוכנת צ'אט עם שיתוף מסודר שתפתור את הבעיה הזאת.
2. במשך כל לימודי הרשתות וההצפנה מעולם לא התעסקנו גם בתחום שמאחורי הקוד, החלק בתוכנה שנראה ללקוח, לכן רציתי לשלב גם את זה בעבודה.  
   במשך כל לימודי הנדסת התוכנה התמקדנו בתקשורת ובעצם בכל מה שקורה "מאחורי המסך". רציתי שחלק עיקרי בפרויקט שלי יהיה ממשק משתמש גרפי (GUI) גם לאור העובדה שאני מתעניינת בעיצוב גרפי.
3. אני מאמינה שסדר וארגון הם הבסיס ליצרנות, ועל כן יש להפוך את שיתוף הקבצים בצ'אט למסודר כדי להפוך אותו לכלי עבודה יעיל לתלמידים, סטודנטים ועובדים.

**הבעיות שהתוכנה פותרת:**

1. תוכנת הצ'אט שהייתה פעם כלי לשליחת מסרונים ותקשורת בלבד, שבה המידע אינו נגיש לאחר ההתנתקות הפכה היום לכלי המועדף לשיתוף קבצים והודעות חשובות. כאשר לקוח מתחבר לתוכנת צ'אט קונבנציונלית הוא יכול מצד אחד לראות את ההודעות האחרונות שנשלחו ומצד שני יש לו גישה לכלל הקבצים ששותפו בתור רשימה ללא הקשר. זה יוצר בעיה כאשר לקוח מעוניין בקובץ ששותף בצ'אט לפני מספר חודשים או אפילו שנים, במיוחד כאשר תאריך השליחה אינו ידוע. לפעמים המשתמש עובר על כל הקבצים בהיסטוריית הצ'אט ולפעמים מבקש שליחה חוזרת של הקובץ ויוצר כפילויות של קבצים. התוכנה שלי מאפשרת שמירה מסודרת של קבצים וכינוי קבצים בשם כך שמציאת קובץ הופכת ממטלה מייגעת לעניין פשוט.
2. חלק מהאנשים בוחרים בלית ברירה בגלל חוסר הנוחות של הצ'אט להשתמש במייל, בלוג או ענן לשיתוף קבצים אך באופן בלתי נמנע נשלחים אליהם קבצים חשובים גם בצ'אטים בהם הם חברים. הם צריכים לשמור את אותם הקבצים מהצ'אט בתוכנה אחרת המועדפת עליהם ועניין זה הוא מסורבל ולא נוח. התוכנה מאפשרת ריכוז של כל הקבצים החשובים במקום אחד.

**משתמשים פוטנציאליים:**

* תלמידים בבתי ספר שקבוצת הכיתה שלהם לא מאפשרת להם שיתוף נוח של קבצים. מצד אחד הפלאפון הנייד הוא הפתרון המידי לצילום הלוח אחרי השיעור או שליחת פתרון תרגילים אך מצד שני תלמיד שיהיה מעוניין באותם הצילומים כמה ימים אחרי יתקשה למצוא אותם.
* סטודנטים וקולגות שמעוניינים בשיתוף יעיל ונוח של קבצים הקשורים ללימודים ולעבודה.
* מורים שמעוניינים לשתף מערכי שיעור וכדומה.
* כל קבוצת אנשים אחרת ששמה לעצמה כמטרה לעבוד על פרויקט משותף.

**מרכיבי התוכנה**

מספר שרתים: שרת משתמשים, שרת צ'אט ושרת קבצים אשר מתקשרים בינם לבין עצמם ובינם לבין הלקוחות.

שרת משתמשים ושרת SQL – אחראי על הרשמה, התחברות, מחיקת משתמשים, עדכון פרטי משתמש, אחזור סיסמאות, סיפוק מידע על משתמשים.

שרת צ'אט – אחראי על העברה מיידית של הודעות וקבצים בין משתמשים בצ'אט פרטי או בקבוצה.

שני קליינטים וסרבר אחד, הקליינטים מתקשרים עם הסרבר והוא אחראי לשליחת ההודעות אל הקליינט השני. הקליינטים מתקשרים אחד עם השני באמצעות הודעות טקסט בלבד, כל שינוי בחלון האפליקציה גם הוא עובר באמצעות הודעת טקסט.

מטרות הפרויקט

**המוצר המוגמר אמור לבצע את הדברים הבאים:**

המוצר המוגמר אמור לתפקד בתור תוכנה המאפשרת מרחב עבודה משותף, מאובטח ונוח בין שני משתמשים. במרחב עבודה זה יכולים המשתמשים לכתוב טקסט בכתבן, לצייר בצייר או לשחק איקס עיגול.

**מטרות מרכזיות:**

לאפשר יצירת קשר בין שני משתמשים רשומים לשרת ולאחר מכן יצירת מרחב עבודה זהה בין שני המשתמשים, כל משתמש יראה בדיוק את מה שרואה המשתמש השני. כל שינוי שייעשה יעשה באחריות המשתמשים, תזוזת עכבר משתמש אחד תיראה גם בחלונו של המשתמש השני (בטופס הראשי), שינוי גודל המסך על ידי אחד המשתמשים ייראה גם בחלון הלקוח השני, פתיחת טופס הכתבן על ידי משתמש אחד למשל תיראה גם אצל חלון המשתמש השני.  
קצב העברת השינויים יהיה מהיר, ללא עיכובים, אחרת עלול להשתבש עיבוד ההודעות.   
בנוסף לכך, הקשר בין המשתמשים לכל אורך הדרך יהיה מוצפן ומאובטח כך שגם אם צד שלישי מאזין לקשר, הוא לא יידע לפענח את ההודעות העוברות.

# שפות התכנות וסביבות העבודה

בפרויקט שלי עשיתי שימוש בשתי שפות תכנות2.7 python, C#, כאשר ב- C# רשמתי את מרבית הפרויקט והוא אחראי על ההצפנה והפענוח, על שליחת המידע דרך הסוקטים, על ממשק המשתמש, כאשר כל מה שכתוב בחלק התוכנית שבפייתון הוא תוכנית הסרבר שמטרתה היא להעביר הודעות את ההודעות שקיבלה אל הקליינט השני.

החלק של הפרויקט בpython- נכתב בעזרת סביבת הפיתוח PyCharm בגרסת 2016.3.3 והחלק של הפרויקט ב-C# נכתב בעזרת סביבת הפיתוח Visual Studio 2015.

הפרויקט נרשם ונבדק רק על מערכת ההפעלה windows 10 64 bit, אולם הוא אמור לעבוד בכל שאר מערכות ההפעלה המודרניות של ווינדוס.

**   /  **

# ניתוח הבעיה האלגוריתמית

הבעיות מסודרות לפי חשיבות וסדר גודל הבעיה.

1. **המרת הודעות טקסט לשינויים חזותיים:**

התקשורת בתוכנה מתבצעת אך ורק על ידי הודעות טקסט מוצפנות, עבור כל שינוי במסך לקוח אחד נשלחת הודעת טקסט שמראה את השינוי על מסך הלקוח השני. מאחר והשינויים מועברים בין הלקוחות על ידי הודעות טקסט יש להמיר את זה אצל הלקוח המקבל לשינוי ויזואלי אותו הוא יכול לראות ולא לקרוא.

1. **הרשמה וכניסה של לקוח למערכת:**

על מנת לעשות כל שימוש בתוכנה שלי, על לקוח חדש להירשם למערכת, ולהפוך למשתמש רשום. הלקוח צריך לשלוח את המידע בנוגע להרשמה שלו (שם משתמש וסיסמה) אל השרת. לאחר מכן, כאשר הלקוח מנסה להתחבר בעזרת המידע איתו הוא נרשם, הוא צריך שוב להעביר את המידע לשרת על מנת לבצע בדיקה מיהו המשתמש שהתחבר (לפי שם המשתמש והסיסמה). לכן, חשוב למצוא דרך להעביר פרטים חשובים אלו אל השרת בבטחה.

1. **יצירת חיבור בין השרת ללקוחות:**

מכיוון שעל מנת לעשות שימוש כלשהו בתוכנה שלנו, על הלקוח להתחבר בעזרת שם המשתמש והסיסמה שלו, ובשביל כך, צריך להעביר את מידע ההתחברות לשרת, וגם מכיוון שלאחר מכן, בשביל לדבר עם משתמשים אחרים בצ'אט יש צורך בעברת מידע מהלקוח ללקוחות אחרים (בעזרת השרת שמתווך ביניהם) יש צורך קודם כל ביצירת חיבור בין הלקוח לשרת בעזרת. השרת מאזין לכל הלקוחות, ולכן צריכה להיות אפשרות ללקוחות רבים להתחבר במקביל לשרת, כך שכל המידע הפרטי העובר בין כל הלקוחות לשרת הוא מוצפן**.**

1. **הצפנת המידע:**

על מנת שהמידע שעובר בין השרת ללקוחות יהיה מאובטח, כך שאף אחד אחר מאשר זה ששלח וזה שקיבל את המידע לא יוכל לקרוא אותו, היה צורך בהצפנת המידע בעת שליחתו מצד לצד. פרטיות הלקוחות היא ערך עליון בתוכנה שלי ולכן היה עליי למצוא דרך בה יוכלו לדבר ללא סכנת האזנה של צד שלישי. הצפנת המידע כוללת יצירת מפתח זהה בשני הלקוחות ללא אופציה שאותו צד שלישי זדוני יידע את המפתח.

1. **שמירת הסיסמאות:**

על מנת לוודא האם הסיסמא שהוקלדה על ידי המשתמש באמת נכונה על השרת לשמור את הסיסמא ולהשוות בינה לבין זאת שהוקלדה. אולם, האם זה בטוח ששרת ישמור אצלו את סיסמאות המשתמשים שלו? לכן, אסור לשמור אצל השרת את הסיסמאות אך הוא חייב לשמור משהו שייצג את הסיסמא.

# תיאור אלגוריתמים קיימים

**המרת הודעות טקסט לשינויים חזותיים:**

* שימוש בסריאליזציה של מבנה נתונים המוגדר מראש בשני הלקוחות, הפיכתו למערך בתים, שליחת מערך הבתים והפיכתו חזרה לאותו מערך נתונים בלקוח השני.
* שליחת מחרוזות מוסכמות מראש כך שהלקוח המקבל מחרוזת מסוימת יידע לאיזה טופס היא מיועדת, הפעולה שנעשתה והפרמטרים שלה לפי סדר ידוע מראש.

**הרשמה וכניסה של לקוח למערכת:**

- הצפנת הסיסמא בלבד.

- הצפנת כל המידע בין השרת ללקוח בחלק הראשוני.

**יצירת חיבור בין השרת ללקוחות:**

* יצירת חיבור רציף ואמין בעזרת socket TCP.
* יצירת חיבור מהיר אך לא רציף ולא אמין בעזרת UDP socket.
* שימוש במחלקה stream (בלקוח) כדי לקרוא ולכתוב הודעות אל השרת.
* שימוש בחיבור SSL בין השרת ללקוח.
* שימוש ב-Threading על מנת להקשיב למספר לקוחות בו זמנית.
* שימוש ב-select, על מנת להקשיב למספר לקוחות בו זמנית.

**שמירת סיסמאות בשרת:**

* שימוש בפונקצית hash שמבצעת פעולות רבות על הסיסמא ככה שעבור כל סיסמא שנקלטת יהיה פלט אחד ויחיד שאי אפשר להחזיר אחורה ולגלות את הסיסמא המקורית.

**הצפנת המידע:**

* שימוש בפרוטוקול SSL כדי ליצור socket מאובטח עם השרת.
* שימוש בהצפנה סימטרית (AES) בלבד להצפנת המידע.

**יצירת מפתח הצפנה סימטרי:**

* שימוש בRSA .
* שימוש באלגוריתם DiffieHellman.

# יתרונות וחסרונות

1. **המרת הודעות טקסט לשינויים חזותיים-**
   1. סריאליזציה:

יתרונות-

סריאליזציה קלה לשימוש.

סריאליזציה נתמכת בטכנולוגיות חדשות וחשובות בתוכנות מודרניות.

חסרונות-

ככל שאובייקט גדול ומורכב יותר כך עולה דרישת הזיכרון מהמערכת כך שאוסף הזבל במערכת עלול להיתקע.

(bשימוש במחרוזות מוסכמות מראש**-**

יתרונות-

דרישת זיכרון נמוכה, החבילות העוברות הן רק המידע הדרוש.

חסרונות-

לא קל לשימוש, מתבסס על עבודה ידנית.

רמת תכנות נמוכה.

1. **הרשמה וכניסת הלקוח למערכת-**
   1. הצפנת כל התקשורת בין השרת ללקוח בחלק זה-

יתרונות-

כל התקשורת תהיה מאובטחת. פחות סכנות של האזנות מצד שלישי.

חסרונות-

הצפנה לא פשוטה (בין שתי שפות שונות).

התוכנה עובדת יותר להצפין ולפענח מידע שהוא בכל מקרה עקבי ברובו בין כל המשתמשים.

* 1. הצפנת הסיסמא בלבד-

יתרונות-

ההודעה החשובה היחידה מחלק זה של התקשורת היא הסיסמא, לכן האבטחה זהה בין שתי הדרכים.

הצפנת ופענוח הודעה אחת במקום הרבה מאפשר חיסכון של עבודת התוכנה.

חסרונות-

שאר ההודעות מחלק זה אינן מוצפנות וצד שלישי יכול לקרוא אותם.

1. **יצירת חיבור בין השרת ללקוחות-**
   1. שימוש בחיבור בפרוטוקול TCP –

יתרונות-

חיבור אמין ועקבי בין הלקוח לשרת, כל מידע שיישלח מהלקוח יתקבל בשרת ולהפך.

חסרונות-

חיבור איטי לעומת האופציה השנייה, (Three-Way-Handshake), דורש מעבר של יותר הודעות.

* 1. שימוש בחיבור בפרוטוקול UDP-

יתרונות-

חיבור מהיר בין הלקוח לשרת, מינימום הודעות מועברות בין הלקוח לשרת כדי ליצור קשר.

חסרונות-

חוסר אמינות, מידע שנשלח עלול לא להתקבל בצד השני.

1. **שמירת סיסמאות בשרת על ידי פונקציות 1Hash-**
   1. שימוש ב SHA256-

יתרונות:

תוצאה של 64 תווים, ארוכה ביחס לרוב הפעולות האחרות.

חסרונות:

איטי יותר ביחס לרוב הפונקציות האחרות.

* 1. שימוש ב MDA5-

יתרונות:

תוצאה של 32 תווים, קצרה ביחס לפעולות אחרות.

חסרונות:

איטי גם כן, לא מהיר כפי שהיינו מצפים מאורך קצר יותר של תוצאה.

**1**נתונים נלקחו מ- <http://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-performance/> . באתר ערכו השוואה בין 4 פונקציות הhash הראשיות.

1. **הצפנת המידע-**
   1. שימוש בפרוטוקול SSL-

יתרונות:

המידע המועבר בsocket מאובטח כך שמי שיכול לקרוא אותו הוא אך ורק הנמען.

חסרונות:

יצירת הקשר והמשכו איטיים יותר משמעותית מהתקשורת הרגילה.

* 1. שימוש בהצפנה סימטרית (AES)-

יתרונות:

המידע המועבר מוגן ומאובטח, מי שיכול לקרוא אותו הוא רק בעל אותו המפתח. ולכן, חשובה העברת מפתח בטוחה.

ההודעות שעוברות הם מערכי בתים רגילים כך שמשתמשים באותו הsocket כמו בחיבור רגיל, המהירות לא מואטת.

חסרונות:

העברת מפתח לא מאובטחת עלולה לגרום לפרצת אבטחה.

1. **יצירת מפתח הצפנה סימטרי-** 
   1. שימוש בהצפנה אסימטרית (RSA)-

יתרונות:

המידע שמועבר ליצירת המפתח יכול להיקרא אך ורק על ידי הלקוח המחזיק במפתח הפרטי.

חסרונות:

מעבר של יותר הודעות, מעבר של המפתח הפומבי הסימטרי.

כל צד שלישי שמשיג את המפתח הפומבי יכול לשלוח הודעות אל הלקוח ובכך לשבש את תהליך יצירת המפתח.

* 1. שימוש באלגוריתם DiffieHellman-

יתרונות:

מעבר של פחות הודעות מאשר הפתרון השני.

על ההודעות שמועברות להיות במבנה מסוים כך שהלקוח מתעלם מהודעות של צד שלישי שלא זהות במבנה.

זה כמעט בלתי אפשרי לזהות את המפתח שנוצר מההודעות שמועברות.

חסרונות:

יצירת המפתח כוללת חישובים מסובכים בין מספרים ארוכים מאוד.

# הפתרון הנבחר

את חלק מההצעות שהוצעו באלגוריתמים הקיימים מימשתי וחלקם לא. כעת אסביר בצורה יותר מעמיקה מדוע בחרתי בכל דרך ואיך היא משפיעה על המבנה של הפרויקט.

**המרת הודעות טקסט לשינויים חזותיים:**

את המרת הודעות הטקסט לשינויים חזותיים בחרתי לעשות על ידי מחרוזות מוסכמות מראש וזאת מכיוון שאחת הבעיות שמפריעות לי בתוכנות הקיימות כיום היא בעיית המהירות. על מנת להיצמד כמה שיותר למהירות זמן האמת בחרתי לעבוד עם מחרוזות פשוטות ולהימנע מסריאליזציה כדי לחסוך את המעבר מאובייקטים ולאחר מכן חזרה אל אותם אובייקטים. כמות ההודעות שעוברת בין הלקוחות דרך השרת לאחר יצירת הקשר היא גבוהה אך מכיוון שההודעות הן מחרוזות פשוטות, המהירות היא מהירה כך שהאיבוד מידע היחידי שקורה הוא בגלל איחוד הודעות הקורה בקבלתן בשרת. איבוד מידע זה הוא זניח וחסר השפעה.

פתרון זה מתבטא בקוד במספר אזורים:

במחלקה Communicator, קולטת את ההודעה ומעבירה את ההודעה לטופס מסוים לפי השם שלו כך שהטופס הוא במקום הראשון של המערך.

///Split the data according to " " eparator

msg\_array = msg.Split(' ');

///Checking to which form the message is destined.

For (receive = 0; receive < openForms.Count; i++)

{

///If the form name equals to a form in the list.

If (msg\_array[0].Equals(openForms[i].Name))

{

///Invoke the Receive method at the specified form.

openForms[i].Recieve(msg\_array);

///Set sent to true.

Sent = true;

}

המקום השני במערך מעובד בטופס המיועד, על ידי הפעולה Receive הקיימת בכל טופס, על ידי switch(msg\_array[1]) למשל, הנה שני מקרים מהטופס mainForm:

switch (msg\_array[1])

{

case "/Window\_size":

{

flag = false;

this.Height = int.Parse(msg\_array[2]); //Set the new window size.

this.Width = int.Parse(msg\_array[3]); //Set the new window size.

}

break;

case "/RequestDiffie":

{

Communicator.SendMessage(this.Name, "/FinalDiffie " + Communicator.GetResponseMessage(msg\_array[2]));

Communicator.SetToClient();

this.NotepadButton.Enabled = true;

this.TiTaToButton.Enabled = true;

} break;

המקומות הבאים במערך, אם ישנם, מעובדים כפרמטרים במקרים מסוימים, למשל במקרה של "/Window\_size", הפעולה מקבלת במקום ה-3 את גובה החלון ובמקום ה-4 את רוחב החלון.

**הרשמת וכניסת לקוח למערכת:**

התקשורת בחלק הראשון של הפרויקט, בעת רישום וכניסת הלקוחות למערכת ועד ליצירת הקשר בין שני לקוחות היא לא מוצפנת מלבד הודעה אחת שהיא הסיסמא. שאר ההודעות בחלק זה הן טכניות ולא מכילות פרטים אישיים לגבי הלקוח מלבד שם המשתמש שלו ולכן לא צריך להצפין אותן. אני מצפין את הסיסמא על ידי יצירת מפתח באלגוריתםDiffieHellman ולאחר מכן, מבצע פעולת XOR של הסיסמא יחד עם המפתח שיצא. את התוצאה אני שולח אל השרת שם מבצע על ההודעה שוב פעם פעלת XOR של ההודעה יחד עם המפתח שיצא. התוצאה מפעולה זו היא הסיסמא.

פתרון זה מתבטא בקוד במספר אזורים:

בעת הכניסה למערכת, השרת והלקוח מממשים את אלגוריתם DiffieHellman על ידי 2 הודעות-

לקוח (חלק מפעולה Connect , Communicator.cs)-

serverDH = new DiffieHellman(256).GenerateRequest(10);

srvr.Send(Encoding.ASCII.GetBytes(serverDH.ToString()));

int bytesrec= srvr.Receive(data);

serverDH.HandleResponse(Encoding.ASCII.GetString(data));

serverKey = new BigInteger(serverDH.Key).ToString();

שרת –

message = data.split(**"|"**)  
server\_dif = DiffieHellman(message[0], message[1])  
shared\_key = str(server\_dif.gen\_shared\_key(message[2]))  
registered\_profiles.append(Profile(new\_socket, shared\_key))  
new\_socket.send(str(server\_dif.gen\_public\_key()))

ולאחר יצירת הקוד המשותף, הצפנת הסיסמא נעשית על ידי פונקצית XOR-

בלקוח-

public static string xorEncryptDecrypt(string input)

{

Console.WriteLine("ServerKey: "+serverKey);

Console.WriteLine("Input: "+ input);

// check for null, empty strings etc.

if (String.IsNullOrEmpty(input) || String.IsNullOrEmpty(serverKey))

return input;

// Do not append string in the loop; use StringBuilder for this

StringBuilder cyphertext = new StringBuilder(input.Length);

// You have no need in SubString and other stuff: just char xor char

// i % pad.Length - if pad is shorter than plaintext

for (int i = 0; i < input.Length; ++i)

cyphertext.Append((char)(input[i] ^ serverKey[i % serverKey.Length]));

Console.WriteLine(cyphertext.ToString());

return cyphertext.ToString();

}

ופונקצית XOR בשרת-

**def** xor\_encrypt\_decrypt(key, input2): output = []  
 **for** i **in** range(len(input2)):  
 xor\_num = ord(input2[i]) ^ ord(key[i % len(key)])  
 output.append(chr(xor\_num))  
  
 return\_str = str(**''**.join(output))  
 return\_str = return\_str[0:return\_str.\_\_len\_\_()-1]  
 **return** return\_str

**יצירת החיבור בין השרת ללקוח:**

את החיבור בין השרת ללקוחות בחרתי לעשות דרך TCP socket על מנת לקבל אמינות בשליחת המידע וקבלת כל המידע בצורה מדויקת בצד השני. לפני יצירת הקשר בין שני הלקוחות, חשוב מאוד שכל הודעה מהלקוח תתקבל בשרת ותשלח חזרה תגובה מתאימה מכיוון שאחרת הלקוח והשרת יהיו במקומות שונים מבחינת ההתקדמות לקראת יצירת הקשר. בנוסף ליתרון קבלת כל המידע, פרוטוקול TCP מבטיח גם שכל המידע היוצא מהלקוח יגיע אל השרת לפי סדר ההגעה. לאחר יצירת הקשר בין שני הלקוחות סדר ההגעה הוא חשוב מאוד מכיוון שאחרת, עלולות להגיע אל אחד הלקוחות הודעות שהוא לא יידע איך לעבד אותן מפני שהודעות שנשלחו קודם עוד לא הגיעו ועברו עיבוד.

הפתרון מתבטא גם אצל השרת וגם אצל הלקוח-

בשרת, חיבור על ידי socket בפרוטוקול TCP (ברירת המחדל)-

s\_socket = socket.socket()  
s\_socket.bind((**"0.0.0.0"**, 11000))  
s\_socket.listen(5)

ובלקוח-

srvr = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

srvr.Connect(remoteEP);

**שמירת סיסמאות בשרת על ידי פונקצית Hash:**

את שמירת הסיסמאות בשרת במבנה נתונים בחרתי לבצע על ידי פונקצית Hash SHA256. בתחילה ניסיתי לבצע תקשורת מוצפנת בין השרת ללקוח לאורך כל הדרך, אך נתקלתי בבעיות להתאים בין פונקציות בפייתון לפונקציות מקבילות בC#. בניסיון זה המפתח הצפנה בסופו של דבר היה SHA256 של המחרוזת שהתקבלה וזאת מכיוון שהדרישה של הפונקציה היה למפתח של 64 בתים (כמו שSHA256 מספק). לכן, דבקתי בפונקציה זו גם לאחר שהניסיון לא הצליח, למטרות אחרות. ובנוסף לכך, בבדיקה2 שפורסמה על ידי מתכנתים באינטרנט, פונקציה זו מספקת את התוצאה הבטוחה ביותר בזמן המהיר ביותר בהשוואה לפונקציות אחרות.

הפתרון בא לידי ביטוי בשרת-

h = hashlib.sha256()

...

h.update(password\_plain) *# Sign in case.***if** h.digest() == called\_member.text:

...

**if** doc.find(username) **is** None: *# Sign up case.* h.update(password\_plain)  
 ET.SubElement(doc, username).text = h.digest()  
 h = hashlib.sha256()

**2**נתונים נלקחו מ- <http://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-performance/> . באתר ערכו השוואה בין 4 פונקציות הhash הראשיות.

**הצפנת המידע:**

על מנת לשמור על אבטחת המידע שעובר בחרתי להצפין את המידע בין הלקוחות על ידי הצפנה סימטרית (AES) וזאת מכיוון שהצפנה סימטרית נעשית על ידי אותו המפתח והיא יעילה יותר מבחינת כמות הודעות שעוברת בין הלקוחות ובזמן הפענוח/הצפנה מאשר הצפנה אסימטרית (RSA) ובנוסף לכך, יצירת הקשר יותר מהירה לעומת שימוש בפרוטוקול SSL למטרות אבטחה.

פתרון זה בא לידי שימוש בקוד על ידי המחלקה Cryptography.cs שהיא מחלקה שהשתמשתי ממקור קוד פתוח מהאינטרנט מומלץ על ידי מתכנתים רבים מהכתובת הזו: <http://stackoverflow.com/a/10177020/5631525>

בקוד אני משתמש בפונקציות מהמחלקה הזו בעת שליחת וקבלת הודעות לאחר יצירת הקשר בין שני הלקוחות-

פענוח-

else // Talking to client- encrypted messages.

{/// Taking only the encrypted data - without the bytes represent length.

stringData = string.Join("", stringData.Skip(4).Take(length).ToArray());

/// Decrypts the data.

msg = Cryptography.Decrypt(stringData, addresseeKey);

}

הצפנה-

Else ///Talking to client - encrypted messages  
{/// Encrypted data.  
byte[] datatosend = Encoding.ASCII.GetBytes(Cryptography.Encrypt(data2, addresseeKey));

///Sends the data and additional first 4 bytes represent the length.

srvr.Send(CombineByteArrays(new byte[][] { GetDataLength(datatosend.Length), datatosend }));

}

**יצירת מפתח להצפנה סימטרית:**

על מנת ליצור את המפתח הסימטרי להצפנה, השתמשתי באלגוריתם של DiffieHellman, אלגוריתם המאפשר הן יצירת מפתח סימטרי והן הצפנה אסימטרית. אלגוריתם זה מאפשר לי ליצור מספר ארוך שכמעט בלתי אפשרי ליצור לפי הפרמטרים אותם אני מעביר (לא מוצפנים) בין הלקוחות, כלומר, לפי פרמטרים אלו רק שני הלקוחות יכולים ליצור מספר גדול מסוים. את המספר הגדול אני ממיר למחרוזת שהיא משמשת לי למפתח הצפנה.

פתרון זה בא לידי שימוש בקוד על ידי המחלקה DiffieHellman.cs ועל ידי מחלקות המשנה BigInteger.cs (לא להתבלבל עם המשתנה BigInteger של System.Numerics, המחלקה כאן הוגדרה בהתאמה אישית לאלגוריתם) והמחלקה StrongNumberProvider.cs, אלו הן מחלקות שהשתמשתי ממקור קוד פתוח מהאינטרנט שהומלץ על ידי מתכנתים מהכתובת הזו: <https://www.codeproject.com/Articles/24632/Shared-Key-Generation-using-Diffie-Hellman>

בקוד, אני משתמש בפונקציות מהמחלקה במספר מקומות, למשל:

בעת התחברות לשרת –

serverDH = new DiffieHellman(256).GenerateRequest(10);

srvr.Send(Encoding.ASCII.GetBytes(serverDH.ToString()));

int bytesrec= srvr.Receive(data);

serverDH.HandleResponse(Encoding.ASCII.GetString(data));

serverKey = new BigInteger(serverDH.Key).ToString();

# דרישות ומוגבלויות המערכת

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

ראשית כל, יש לוודא שכל המודולים החיצוניים שמוזכרים בביבליוגרפיה של עבודה זו (אחרי תיעוד הקוד ולפני הקוד עצמו).

למערכת כמה מוגבלויות והן כדלהלן:

* אני מניח שהלקוח יודע את הip והport שבו פועל השרת.
* בכתבן ניתן לרשום רק בשפה האנגלית (כולל מספרים וסימנים רגילים).
* בכתבן ניתן לרשום מקסימום של 4060 תווים בשל הגבלת גודל הפקטה.
* בעת השימוש בכתבן, תזוזות מהירות מידיי של העכבר יגרמו לקווים ישרים במקום לעגולים.
* בעת יצירת הקשר בין הלקוחות, למערכת עלול לקחת מספר שניות להתחיל להעביר הודעות וזאת מכיוון שהלקוחות מסכימים ביניהם על מפתח הצפנה משותף. כל הכפתורים אינם זמינים ללקוח עד שיוסכם המפתח.
* במידה והסרבר נסגר ידנית הלקוחות אינם נשמרים רשומים למערכת ולכן עליהם להירשם מחדש.
* לא ניתן לשנות את גודל החלון בטופס Paint וזאת מכיוון ששינויו יפגע בציור שנצבע.

# תיאור ממשק המשתמש והמודולים

תרשים זרימת ממשק המשתמש נראה כך:

**לחיצה על כפתור ואישור המשתמש השני**

**המשתמש הכניס שם משתמש וסיסמא נכונים**

**אם המשתמש עוד לא רשום**

**תקשורת מוצפנת דרך socket**

**כפתור**

**כפתור**

עץ מודולים עבור צד הלקוח:

firstForm.cs

**תבנית מחלקה של Observer**

**קשר עם השרת**

**פותח את התהליך**

צד השרת

Communicator.cs

Notepad.cs

TicTacToe.cs

mainForm.cs

SignUpForm.cs

SessionForm.cs

loginForm.cs

Paint.cs

# פירוט המודולים בלקוח:

firstForm.cs:

תחילת ריצת התוכנית של הלקוח, הטופס הראשון שעולה בו הלקוח מכניס את פרטי הרשת של השרת, IP וport ומנסה להתחבר באמצעות פרטים אלו.

loginForm.cs:

טופס התחברות של המשתמשים למערכת על ידי שם משתמש וסיסמא הנבדקים בשרת. כדי להירשם למערכת עליו ללחוץ על כפתור ההרשמה וייפתח הטופס SignUpForm.cs. אם רשום, ינסה להתחבר ובמידה וצדק בפרטיו ייפתח הטופס SessionForm.cs.

SignUpForm.cs:

טופס הרשמה של לקוח למערכת, הטופס אחראי להעביר את פרטי המשתמש לשרת ולקבל תגובה. אם הפרטים מתאימים, הלקוח יוחזר לטופס ההתחברות. אחרת, יישאר בטופס ויקבל הודעה מתאימה.

SessionForm.cs:

טופס התקשרות של הלקוח ללקוח אחר, הטופס אחראי ליצירת הקשר בין שני לקוחות לפי בחירתם. במידה והצליחו, יועברו הלקוחות אל הטופס mainForm.cs.

mainForm.cs:

הטופס הראשי של המערכת, לאחר שהתקשרו שני הלקוחות טופס זה הוא תחילת החלק בתוכנה שמסונכרן בין שני הלקוחות. לחיצה אצל אחד הלקוחות על כפתור הכתבן תפתח את טופס הNotepad.cs אצל שניהם. לחיצה אצל אחד הלקוחות על כפתור האיקס-עיגול תפתח את טופס הTicTacToe.cs אצל שניהם.

Notepad.cs:

טופס במערכת המייצג כתבן, על הטופס להראות לכל לקוח הן את הקלדותיו והן את הקלדות הלקוח השני.

TicTacToe.cs:

טופס במערכת המייצג איקס עיגול, על הטופס לדמות בין שני המשתמשים משחק איקס עיגול. בסוף המשחק יוכרז המנצח אצל שני הלקוחות וייסגר הטופס. הלקוחות יוכלו להתחיל משחק חדש במידה וירצו.

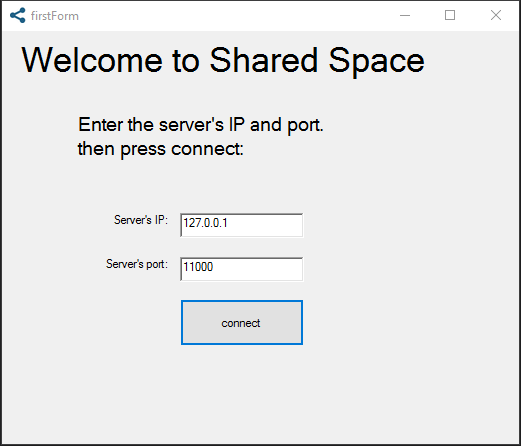
Paint.cs:

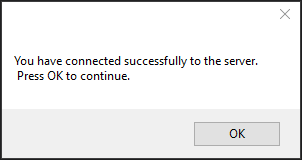
טופס במערכת המייצג צייר, על הטופס לשמש כבלוק ציור. כל לקוח יוכל לצייר במקביל על בלוק הציור בצבע שבחר (צבע ברירת מחדל- שחור). כל לקוח יוכל למחוק ולנקות את כל הלוח.

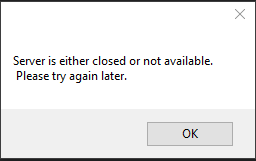
# תיאור הממשק למשתמש – הוראות הפעלה

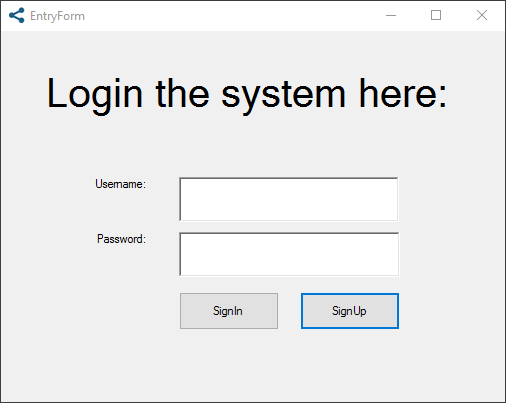
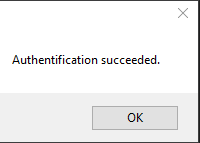
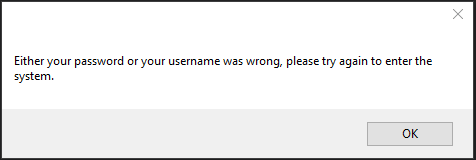
בצד של השרת אין GUI, ועל מנת להריץ את השרת מה שצריך לעשות הוא לוודא שהמודולים השונים קיימים ומסודרים במיקום הנכון. בנוסף צריך לוודא שהשרת רץ על מערכת ההפעלה windows 10 ו-python 2.7. לאחר מכן כל שיש לעשות הוא פשוט להריץ את Server.py.

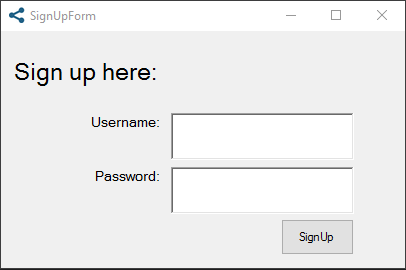
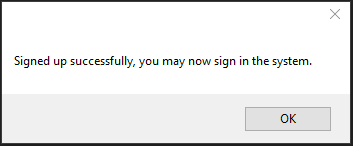
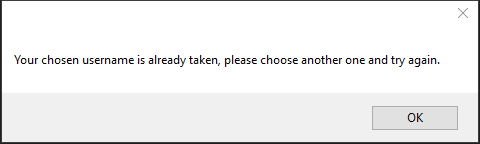
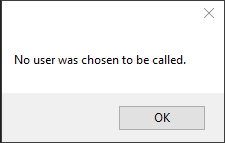
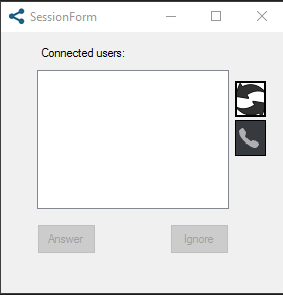
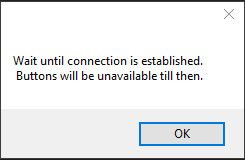
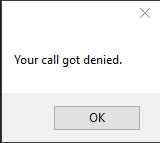
בצד של הלקוח:

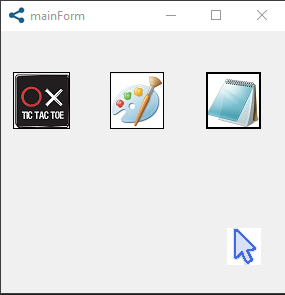
1. יש להריץ את firstForm.cs שפותח את טופס ההתחברות אל השרת על ידי הקלדת פרטי השרת ולאחר מכן ללחוץ על הכפתור Connect.

  
  
  
  
  
  
  
  
  
  
  
  
  
  
במידה והחיבור הצליח תיזרק ההודעה המתאימה ויעלה הטופס הבא-

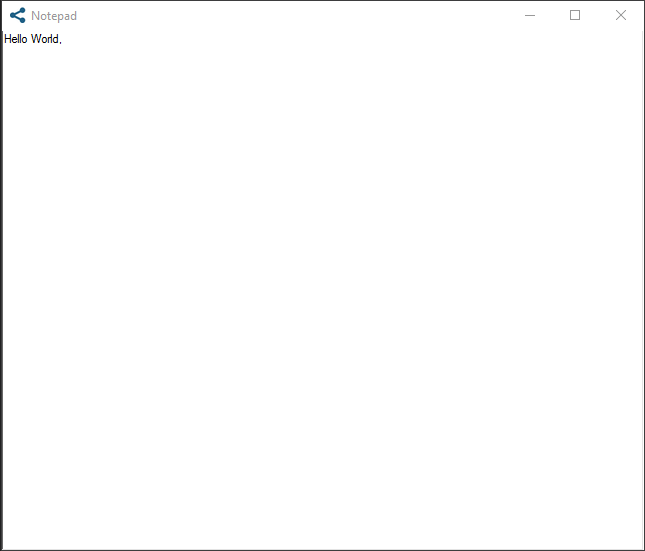
במידה והחיבור לא הצליח תיזרק ההודעה המתאימה ויישאר טופס ההתחברות-   


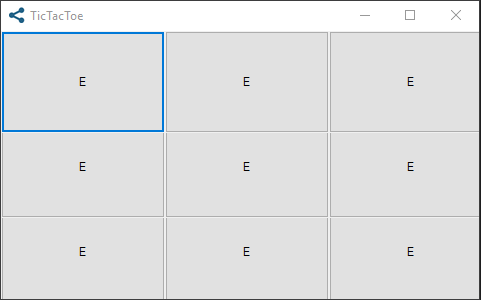
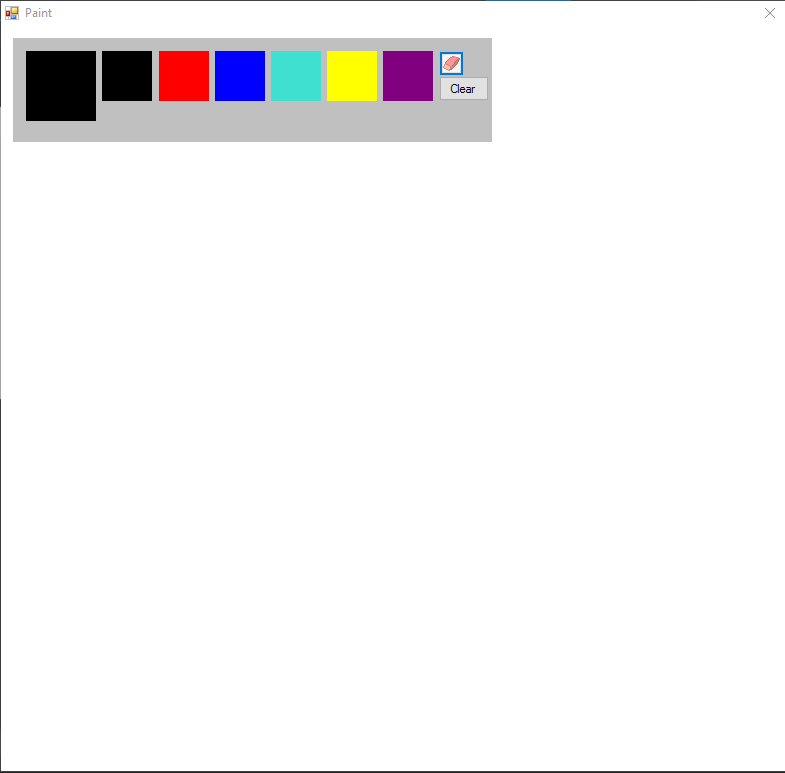
1. לאחר מכן, יקפוץ הטופס הבא- loginForm שיבקש מהמשתמש להתחבר אל השרת-  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   במידה והלקוח קיים במערכת והצליח להתחבר תיזרק ההודעה המתאימה ויעלה הטופס הבא-   
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   במידה והלקוח לא קיים במערכת תיזרק ההודעה המתאימה ויישאר בטופס הזה-  
   

1. במידה והלקוח לא קיים במערכת, עליו להירשם קודם על ידי לחיצת הכפתור sign up והירשמות בטופס SignUpForm:  
     
     
     
   במידה והלקוח הצליח להירשם, תיזרק ההודעה המתאימה והוא יוחזר לטופס הקודם בו יתבקש להתחבר-   
     
     
     
   במידה והלקוח לא הצליח להירשם, תיזרק ההודעה המתאימה-
2. לאחר ההתחברות למערכת יעלה הטופס SessionForm, בו הלקוח ילחץ על הכפתור 'רענן' כדי לרענן את רשימת הלקוחות המחוברים כעת לשרת. מתוך רשימה זו יבחר המשתמש את המשתמש אליו רוצה להתקשר, וילחץ על כפתור ה'התקשר' כדי להתחיל שיחה-  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   במידה והלקוח ניסה להתחיל שיחה כאשר עוד לא בחר משתמש אליו רוצה להתקשר, תיזרק ההודעה המתאימה-  
     
     
     
     
     
     
     
     
     
     
     
     
   במידה והמשתמש התקשר למשתמש אחר והמשתמש האחר ענה לו, תיזרק ההודעה המתאימה-  
     
     
     
     
     
     
     
     
     
     
     
     
   במידה והמשתמש התקשר למשתמש אחר והמשתמש האחר סירב לשיחה, תיזרק ההודעה המתאימה-  
   
3. לאחר התחלת הקשר בין הלקוחות, יעלה הטופס הראשי (mainForm) המשמש כפלטפורמה לכל האפליקציות במערכת.



1. במידה ובחר אחד הלקוחות להפעיל את הכתבן, יופעל הטופס Notepad, בו יכול כל אחד מן הלקוחות לכתוב ולמחוק.



1. במידה ובחר אחד הלקוחות להפעיל את האיקס עיגול, יופעל הטופס TicTacToe שידמה משחק איקס עיגול בין שני הלקוחות:
2. במידה ובחר אחד הלקוחות להפעיל את הצייר, יופעל הטופס Paint, בו יוכל כל אחד מן הלקוחות לצייר כרצונו, למחוק ולנקות את הלוח במידת הצורך.   
   

# השוואה עם פתרונות קיימים

במהלך הכנת הפרויקט עלו בדמיוני רעיונות רבים לביצוע בפרויקט אולם מגבלת הזמן ורמת התכנות הגבוהה הדרושה לביצועם אינן מאפשרות לי. מספר תוספות שרציתי להוסיף לפרויקט הן:

שמירה בטוחה של פרופילי המשתמשים (שמות משתמש וסיסמא) במבנה נתונים מאובטח כמו SQL , שתאפשר גיבוי לשרת במידה וצריך לכבותו למטרות עדכון ושיפור.

התוכנה שהכנתי מהווה פלטפורמה לעוד משחקים או יישומים שונים שבשבילם דרושים שני שחקנים ולכן אפשר להוסיף לפרויקט אפליקציות בנוסף לכתבן, לצייר ולאיקס-עיגול כמו דמקה, שחמט, צוללות, עורך מסמכים, צ'אט ועוד.

הגבלת כמות המשתמשים הנרשמים למערכת על ידי שליחת דואר זיהוי לאימייל של כל משתמש נרשם. במידה ואישר את יצירת המשתמש, לא יהיה ניתן יותר להשתמש בכתובת מייל זו.

# תיעוד הקוד

צד השרת-

Server.py-

| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| --- | --- | --- | --- |
| **class** Client(): | --------- | --------- | מחלקה המחזיקה בשני חוטים . |
| **def** \_\_init\_\_(self, ReceivingThread, SendingThread): | פעולה המקבלת שני חוטים, אחד המקבל הודעות ואחד ששולח הודעות. | פעולה מאתחלת אובייקט מסוג לקוח. |  |
| **def** close(self): | --------- | פעולה הסוגרת את שני החוטים. |  |
| **def** \_\_run\_\_(self): | ---------- | פעולה המריצה את שני החוטים. |  |
| **def** get\_isclosed(self): | ------- | פעולה מחזירה האם הקשר אם לקוח מסוים סגור או לא. |  |
| **class** SendingThread(th.Thread): |  |  | מחלקה שמטרתה הוא להעביר הודעות אל קליינט מסוים. |
| **def** \_\_init\_\_(self, client): | פעולה המקבלת משתנה מסוג סוקט שהוא אחד הלקוחות. | פעולה המאתחלת חוט מסוים. |  |
| **def** run(self): | ---------- | פעולה ששולחת כל הודעה המיועדת ללקוח מסוים. |  |
| **class** ReceiveingThread(th.Thread): |  |  | מחלקה שמטרתה היא לקבל הודעות מקליינט מסוים. |
| **def** \_\_init\_\_(self, client): | פעולה המקבלת משתנה מסוג סוקט שהוא אחד הלקוחות. | פעולה המאתחלת חוט מסוים. | ------- |
| **def** run(self): | ------- | פעולה שתמיד מקבלת הודעות מקליינט מסוים. | ------ |
| **class** DiffieHellman: | ------- | ---------- | מחלקה המממשת את אלגוריתם דיפי-הלמן. |
| **def** \_\_init\_\_(self, new\_p, new\_g): | פעולה מקבלת שני מספרים. | פעולה יוצרת משתנה מסוג דיפי-הלמן. כאשר P הוא מספר ראשוני, G הוא מספר יוצר שלו ו A הוא המפתח הפרטי של השרת. | -------- |
| **def** get\_private\_key(self): | -------- | פעולה מחזירה את A, המפתח הפרטי של השרת. |  |
| **def** gen\_public\_key(self): | ------ | פעולה מחזירה את  *שהוא המפתח הפומבי של השרת.* | ------ |
| **def** gen\_shared\_key(self, other\_contribution): | פעולה מקבלת את  שהוא המפתח הפרטי של השרת. | פעולה מחזירה את המפתח המשותף שהוא – | -------- |
| **class** Profile: | ------- | -------- | מחלקה המייצגת פרופיל של לקוח הרשום לשרת. |
| **def** \_\_init\_\_(self, client, gaxy): | פעולה מקבלת את ה-socket ואת המפתח הצפנה שלו gaxy. | פעולה יוצרת אובייקט מסוג פרופיל. | --------- |
| **def** get\_socket(self): | -------- | פעולה המחזירה את הsocket של פרופיל מסוים. | ------- |
| **def** get\_key(self): | ------- | פעולה המחזירה את המפתח הצפנה של פרופיל מסוים. | ------- |
| **def** get\_username(self): | ------- | פעולה המחזירה את השם המשתמש של פרופיל מסוים. | ------- |
| **def** is\_username\_set(self): | -------- | פעולה המחזירה אמת אם הוגדר שם משתמש של פרופיל מסוים ושקר אחרת. | --------- |
| **def** set\_username(self, username): | פעולה המקבלת מחרוזת. | פעולה ומשנה את תכונת שם המשתמש של פרופיל מסוים למחרוזת שהתקבלה. |  |
| **def** get\_connection\_status(self): | ------- | פעולה מחזירה אמת אם פרופיל מסוים מחובר למערכת ושקר אחרת. |  |
| **def** set\_connection\_status(self, boolean): | פעולה מקבלת משתנה בוליאני. | פעולה משנה את סטטוס החיבור של פרופיל מסוים למשתנה הבוליאני שהתקבל. |  |
| **class** SessionThread(th.Thread): | ------- | ------- | מחלקה המתארת קשר בין שני לקוחות. |
| **def** \_\_init\_\_(self, client1, client2): | פעולה מקבלת משתנה מסוג socket. | פעולה מאתחלת משתנה מסוגsessionThread | ------ |
| **def** run(self): | ------- | ---- | פעולה רצה כל עוד הקשר עם שני הלקוחות פתוח וסוגרת את הקשר עם לקוח אחד כאשר עם השני נסגר. |
| **def** xor\_encrypt\_decrypt(key, input2): | פעולה המקבלת שתי מחרוזות. | פעולה מחזירה את XOR של שתי המחרוזות יחד. | פעולה זו משמשת להצפנה או לפענוח של הסיסמא. |
| **def** set\_profile\_username(client, username): | פעולה המקבלת משתנה מסוג socket ומחרוזת. | פעולה משנה את שם המשתמש של פרופיל המחזיק באותו ה socket כמו הsocket שהתקבל. | -------- |
| **def** set\_connection\_status(client, boolean): | פעולה המקבלת משתנה מסוג socket ומשתנה בוליאני. | פעולה משנה את סטטוס החיבור של פרופיל המחזיק באותו הsocket כמו הsocket שהתקבל. | -------- |
| **def** is\_user\_loggedin(username): | פעולה המקבלת משתנה מסוג מחרוזת. | פעולה מחזירה משתנה מסוג בוליאני שהוא סטטוס החיבור של פרופיל בעל אותו שם משתמש כמו המחרוזת שהתקבלה. |  |
| **def** get\_profile\_key\_by\_socket(socket): | פעולה המקבלת משתנה מסוג socket. | פעולה מחזירה מפתח הצפנה של פרופיל מסוים המחזיק באותו הsocket כמו הsocket שהתקבל. |  |
| **def** get\_profile\_key\_by\_username(username): | פעולה המקבלת מחרוזת. | פעולה מחזירה מפתח הצפנה של פרופיל המחזיק באותו שם משתמש כמו המחרוזת שהתקבלה. |  |
| **def** get\_profile\_socket(username): | פעולה המקבלת מחרוזת. | פעולה מחזירה socket של פרופיל המחזיק באותו שם משתמש כמו המחרוזת שהתקבלה. |  |
| **def** get\_profile\_username(client): | פעולה המקבלת משתנה מסוג socket. | פעולה מחזירה שם משתמש של פרופיל המחזיק באותו socket כמו socket שהתקבל. |  |
| **def** send\_message(client, message): | פעולה המקבלת משתנה מסוג socket ומחרוזת. | פעולה שולחת לsocket שהתקבל את ההודעה שהתקבלה בתוספת 4 בתים המייצגים את האורך שלה. |  |
| **def** get\_data\_length(data\_length): | פעולה המקבלת מספר שלם. | פעולה מחזירה מחרוזת באורך 4 בתים המייצגת את המספר שהתקבל. |  |
| **def** main(): | ------- | פעולה יוצרת את השרת, מקשיבה ללקוחות ומגיבה בהתאם עד ליצירת הקשר בין שני לקוחות. |  |

צד הלקוח-

**Communicator.cs-**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public class Communicator | --------- | --------- | מחלקה שמטרתה היא ליצור ולנהל את הקשר עם השרת. |
| static Communicator( ) { } | --------- | --------- | בנאי סטטי של המחלקה. |
| public static Communicator Instance | --------- | פעולה זו יוצרת מופע סטטי של המחלקה אם לא נוצר כבר מופע כזה. | תבנית סינגלטון. |
| public static void SendMessage(string sender, string message**)** | פעולה מקבלת שתי מחרוזות, האחת היא שם הקורא לפעולה והשנייה היא הודעה. | פעולה שולחת את ההודעה אל השרת. | פעולה זו נקראת על ידי כל הטפסים כאשר רוצים לשלוח הודעות. |
| public static void SendMessage(string message) | פעולה מקבלת מחרוזת. | פעולה שולחת הודעה אל השרת. | פעולה זו נקראת על ידי כל הטפסים כאשר רוצים לשלוח הודעות המיועדות לשרת. |
| public static void RecievingMessage() |  | פעולה זו מקשיבה תמיד להודעות המגיעות מהשרת ואחראית להעביר את ההודעות לטופס אליו הן מיועדות. |  |
| public static void Connect(IPAddress ipAddress, int port) | פעולה זו מקבלת משתנה מסוג IPAddress שמייצג את כתובת הIP של השרת ומשתנה מסוג Int שמייצג את הפורט אליו מאזין השרת. | פעולה זו יוצרת את הקשר עם השרת. | נקראת על ידי לחיצה על הכפתור Connect מהטופס הראשי. |
| internal static byte[] CombineByteArrays(byte[][] arrays) | פעולה שמקבלת מערך של מערכי בתים. | הפעולה מחזירה מערך בתים שהוא שרשור של כל מערכי הבתים מהמערך הנקלט. |  |
| internal static byte[] GetDataLength(int dataLength) | פעולה שמקבלת מספר שלם. | פעולה מחזירה מערך בתים באורך 4 שמכיל את המספר השלם. |  |
| public static void AddForm(baseForm newForm) | פעולה מקבלת משתנה מסוג baseForm. | הפעולה מוסיפה את המשתנה לרשימת הטפסים הפתוחים. | פעולה זו נקראת כאשר טופס חדש נפתח. |
| public static void RemoveForm(baseForm removedForm) | פעולה מקבלת משתנה מסוג baseForm. | הפעולה מסירה את המשתנה מרשימת הטפסים הפתוחים. | פעולה זו נקראת כאשר טופס קיים נסגר. |
| public static DiffieHellman GetRequestMessage() | --------- | פעולה מחזירה משתנה מסוג דיפי-הלמן המכיל בקשה שכוללת בתוכה מספר ראשוני, בסיס ואת הפרמטר הבא- |  |
| public static DiffieHellman GetResponseMessage(string request) | פעולה מקבלת מחרוזת. | פעולה שומרת את מפתח ההצפנה ומחזירה משתנה מסוג דיפי-הלמן המכיל את התגובה לבקשה שכוללת את הפרמטר הבא- |  |
| public static void HandleResponse(string response) | פעולה מקבלת מחרוזת. | פעולה מחשבת ושומרת את מפתח ההצפנה. |  |
| public static string xorEncryptDecrypt(string input) | פעולה מקבלת מחרוזת. | פעולה מחזירה מחרוזת מוצפנת על ידי פונקציית XOR של המחרוזת שהתקבלה והמחרוזת שהתקבלה קודם לכן מאלגוריתם דיפי-הלמן. |  |
| public static bool IsConnectedToServer() | --------- | פעולה מחזירה אמת אם יש חיבור אל השרת ושקר אחרת. |  |
| public static void SetToClient() |  | פעולה משנה את המשתנה הבוליאני שאומר אל מי הלקוח מדבר כעת (אמת- לשרת, ושקר- ללקוח אחר). |  |
| public static void DisConnect() |  | פעולה מנתקת את הקשר עם השרת בבטחה. |  |

**baseForm.cs-**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| **public class baseForm: Form** | --------- | --------- | מחלקה שמטרתה היא שממנה יירשו שאר הטפסים וכך אפשר להגדיר פעולות משותפות לכל הטפסים. |
| **public virtual void Recieve(string[] message)** | פעולה המקבלת מחרוזת. | ---------- | פעולה זו תמומש באופן שונה עבור כל טופס שיעבד את ההודעה המיועדת אליו. |
| **private void InitializeComponent()** | ------- | פעולה זו מאתחלת את המחלקה. | --------- |

**Firstform.cs-**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class firstForm : Form | ------ | ------ | מחלקה המייצגת טופס שבו הלקוח מתחבר אל השרת. |
| public firstForm() | ----- | בנאי של הטופס. |  |
| private void firstForm\_Load(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שהטופס נטען והוא מאתחל את המשתנים של הטופס. |  |
| private void connectButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שכפתור 'התחבר' נלחץ. |  |

**loginForm.cs-**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class loginForm : baseForm |  |  | מחלקה שמייצגת טופס בו הלקוח מתחבר אל השרת. |
| public loginForm() |  | פעולה יוצרת טופס מסוג loginForm. |  |
| private void EntryForm\_Load(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שהטופס נטען והוא מאתחל את המשתנים של הטופס. |  |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומבצעת פקודות עבור כל מקרה. |  |
| private void SignUpButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה פותחת את הטופס signUpForm כל פעם שנלחץ כפתור sign up. |  |
| private void SignInButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה מנסה להתחבר אל המערכת עם הקלט שהוכנס על ידי המשתמש. |  |
| public void SignUpForm\_closed() | -------- | פעולה רצה כדי לסגור את טופס הsign up Form הפתוח. |  |
| private void LoginForm\_FormClosed(object sender, FormClosedEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה המייצגת אירוע המתאר את סגירת הטופס, שמתנתק בבטחה מהשרת. |  |

**SignUpForm.cs-**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class SignUpForm : baseForm |  |  | מחלקה המייצגת טופס בו יכול הלקוח להרשם למערכת וליצור לעצמו משתמש. |
| public SignUpForm(loginForm lf) | פעולה מקבלת משתנה מסוג loginForm. | פעולה יוצרת טופס signupForm ומאתחלת את המשתנים שלו. |  |
| private void SignUpButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה מתארת אירוע בו נלחץ כפתור הsignup. הפעולה מנסה להרשם למערכת עם הנתונים שהוקלדו על ידי הלקוח. |  |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומגיבה בהתאם לכל קלט. |  |
| private void SignUpForm\_FormClosing(object sender, FormClosingEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג FormClosingEventArgs. | פעולה מתארת אירוע בו הטופס נסגר. |  |

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית  **SessionForm.cs-** |
| public partial class SessionForm : baseForm |  |  | מחלקה המייצגת טופס בו הלקוח יוצר קשר עם לקוח אחר המחובר למערכת. |
| public SessionForm() |  | פעולה יוצרת את הטופס ומאתחלת את המשתנים שלו. |  |
| private void SessionForm\_Load(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שהטופס נטען והוא מאתחל את המשתנים של הטופס. |  |
| private void listView1\_SelectedIndexChanged(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שמשתנה הבחירה של הלקוח מתוך רשימת הלקוחות המוצגת בפניו. |  |
| private void refreshButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שנלחץ כפתור 'רענן' ומבקש מהשרת לעדכון הרשימה. |  |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומגיבה בהתאם לכל קלט. |  |
| private void callButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שנלחץ כפתור 'התקשר' ומתקשר אל הלקוח הנבחר. |  |
| private void ignoreCallButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שנלחץ כפתור 'התעלם' וזה אומר שהלקוח לא עונה לשיחה שהתקשרו אליו. |  |
| private void answerCallButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שנלחץ כפתור 'ענה' וזה אומר שהלקוח ענה לשיחה והקשר עומד להתחיל. |  |
| private void SessionForm\_FormClosed(object sender, FormClosedEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג FormClosedEventArgs. | אירוע זה נקרא ברגע שהטופס נסגר, הפעולה מתנתקת מהשרת בבטחה. |  |

**mainForm.cs**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class mainForm : baseForm | --------- | --------- | מחלקה המייצגת טופס שהוא הטופס הראשי ממנו יהיה ניתן להפעיל את האפליקציות במערכת. |
| public mainForm() | ------- | פעולה מאתחלת את הטופס. |  |
| private void mainForm\_Load(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה מאתחלת את המשתנים של הטופס. | פעולה זו נקראת כאשר הטופס נטען. |
| private void mainForm\_MouseMove(object sender, MouseEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג MouseEventArgs. |  | פעולה זו נקראת עבור כל תזוזה של העכבר. |
| private void NotepadButton\_click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה זו פותחת את הכתבן. | פעולה זו נקראת כאשר לוחצים על כפתור ה Notepad. |
| private void mainForm\_FormClosing(object sender, FormClosingEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג FormClosingEventArgs. |  | פעולה זו נקראת כאשר סוגרים את הטופס הנ"ל. |
| private void TicTacToeButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה זו פותחת את האיקס עיגול. | פעולה זו נקראת כאשר לוחצים על כפתור ה TicTacToe. |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומגיבה בהתאם לכל קלט. | פעולה זו נקראת על ידי ה Communicator. |
| private void mainForm\_Resize(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה זו מייצגת אירוע שמתאר שינוי בגודל החלון של הטופס. |  |

**Notepad.cs**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class Notepad : baseForm | --------- | --------- | מחלקה היורשת מbaseForm והיא טופס הכתבן. |
| public Notepad() | ------- | הפעולה מאתחלת את הטופס. |  |
| private void Notepad\_Resize(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה זו מייצגת אירוע שמתאר שינוי בגודל החלון של הטופס. | הפעולה נקראת כאשר הלקוח משנה את הגודל של הטופס. |
| private void richTextBox1\_TextChanged( object sender, EventArgs e ) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. |  | הפעולה נקראת כאשר הלקוח כותב או מוחק תווים מהכתבן. |
| private void Notepad\_FormClosing(object sender, FormClosingEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג FormClosingEventArgs. |  | הפעולה נקראת כאשר טופס הכתבן נסגר על ידי הלקוח. |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומגיבה בהתאם לכל קלט. | פעולה זו נקראת על ידי ה Communicator. |

**TicTacToe.cs**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class TicTacToe : baseForm | --------- | --------- | מחלקה המייצגת טופס שבו יכולים הלקוחות לשחק ביניהם איקס עיגול. |
| public TicTacToe(TicValue newid) | פעולה מקבלת ערך מסוג TicValue שיכול להכיל איקס, עיגול או את האות E שתסמן משבצת ריקה (Empty). | הפעולה מאתחלת את הטופס. |  |
| private void TicTacToe\_Load(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | הפעולה מגדירה משתנים עבור הקוד של משחק האיקס עיגול. | הפעולה נקראת כאשר הטופס עולה. |
| private void button\_click(object sender , EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | הפעולה משנה את הערך של הכפתור לפי זהות הלוחץ וקוראת לפעולה שמבצעת את תורו של השחקן. | פעולה הנקראת כאשר נלחץ כפתור מסוים בלוח (כל משבצת היא כפתור). |
| private void Enablers(bool x) | פעולה מקבלת משתנה בוליאני. | הפעולה משנה את תכונת הפעילות של הכפתורים בלוח true/false לפי המשתנה הבוליאני. |  |
| private void Build() | ----------- | הפעולה מגדירה מטריצת כפתורים ומאפסת את ערכיהם לTicValue המסמן משבצות ריקות. |  |
| private void SetSize(int Height, int Width) | הפעולה מקבלת שני מספרים שלמים (אורך ורוחב). | הפעולה משנה את גודל כל הכפתורים בלוח לפי ערכי המשתנים שקיבלה. |  |
| private void SetLocation() | ---------- | הפעולה משנה את מיקום הכפתורים בלוח לפי גודלו. |  |
| private void TicTacToe\_Resize(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | הפעולה קוראת לפעולות שינוי המיקום והגודל של הכפתורים. | הפעולה נקראת כאשר משתנה גודל הלוח. |
| public void Turn(TicValue currID, int row, int column) | פעולה מקבלת ערך מסוג TicValue שמסמן את זהות הלקוח ושני מספרים המסמלים את מיקום הכפתור עליו לחץ. | הפעולה משנה את ערך הכפתור עליו לחץ. | הפעולה נקראת בעת ביצוע תור של כל לקוח. |
| public bool IsWinRow(int x) | פעולה המקבלת מספר שמסמן מספר של שורה. | הפעולה מחזירה true אחד הלקוחות ניצח עבור השורה המסוימת, אחרת מחזירה false. |  |
| public bool IsWinColumn(int x) | פעולה המקבלת מספר שמסמן מספר של עמודה. | הפעולה מחזירה true אחד הלקוחות ניצח עבור העמודה המסוימת, אחרת מחזירה false. |  |
| public bool IsWinDiagonal(int x) | פעולה המקבלת מספר שמסמן מספר של אלכסון, 0 עבור אלכסון ראשי ו1 עבור אלכסון משני. | הפעולה מחזירה true אחד הלקוחות ניצח עבור האלכסון המסוים, אחרת מחזירה false. |  |
| public bool IsFinished() | ------- | פעולה הבודקת האם המשחק נגמר, מחזירה true אם כן, false אחרת. |  |
| public string WhoWon() | ------- | הפעולה מחזירה מחרוזת שמסמנת מי ניצח במשחק. |  |
| private void TicTacToe\_FormClosing(object sender, FormClosingEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג FormClosingEventArgs. |  | הפעולה נקראת כאשר הלקוח לחץ לסגירת הטופס. |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומגיבה בהתאם לכל קלט. | פעולה זו נקראת על ידי ה Communicator |

**Paint.cs**

|  |  |  |  |
| --- | --- | --- | --- |
| חתימת הפונקציה | תיעוד פרמטרים | ערך חזרה או אפקט | דרך פעולה עיקרית |
| public partial class Paint : baseForm |  |  | מחלקה המייצגת טופס שבו יכולים הלקוחות להשתמש בצייר שיראה את אותו הלוח אצל שניהם. |
| public Paint() |  | פעולה יוצרת טופס מסוג Paint ומאתחלת את הנתונים שלו. |  |
| public override void Recieve(string[] msg\_array) | פעולה מקבלת מערך מחרוזות המייצג את הקלט מופרד על ידי רווח. | פעולה מעבדת את הקלט שהתקבל ומגיבה בהתאם לכל קלט. | פעולה זו נקראת על ידי ה Communicator |
| private void PaintForm\_load(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא ברגע שהטופס נטען והוא מאתחל את המשתנים של הטופס. |  |
| private void DefaultColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע ברירת המחדל ומשנה את צבע העט לשחור. |  |
| private void RedColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע האדומה ומשנה את צבע העט לאדום. |  |
| private void BlueColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע הכחולה ומשנה את צבע העט לכחול. |  |
| private void GreenColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע שבצבע טורקיז ומשנה את צבע העט לטורקיז. |  |
| private void YellowColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע הצהובה ומשנה את צבע העט לצהוב. |  |
| private void PurpleColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע הסגולה ומשנה את צבע העט לסגול. |  |
| private void PaintForm\_MouseDown(object sender, MouseEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג MouseEventArgs. | אירוע זה נקרא בעת לחיצה על העכבר. |  |
| private void PaintForm\_MouseUp(object sender, MouseEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג MouseEventArgs. | אירוע זה נקרא בסוף הלחיצה על העכבר. |  |
| private void PaintForm\_MouseMove(object sender, MouseEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג MouseEventArgs. | אירוע זה נקרא בעת תזוזת העכבר. במידה והעכבר לחוץ הוא יצייר על בלוק הציור. |  |
| private void BringAllToFront() |  | פעולה המזיזה את כל תיבות הצבע והכפתורים לקדמת המסך. |  |
| private void EraserButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת הלחיצה על כפתור ה'מחק'. פעולה זו מחליפה את צבע העט ללבן בעובי עבה מהעובי הרגיל. |  |
| private void ClearButton\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת הלחיצה על כפתור ה'נקה'. פעולה זו מנקה את כל המסך. |  |
| private void PaintForm\_Resize(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | פעולה זו מייצגת אירוע שמתאר שינוי בגודל החלון של הטופס. | הפעולה נקראת כאשר הלקוח משנה את הגודל של הטופס. |
| private void BlackColorBox\_Click(object sender, EventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג EventArgs. | אירוע זה נקרא בעת לחיצה על תיבת הצבע השחורה ומשנה את צבע העט לשחור. |  |
| private void PaintForm\_FormClosing(object sender, FormClosingEventArgs e) | פעולה המקבלת שני משתנים, האחד מסוג object והשני מסוג FormClosingEventArgs | פעולה מתארת אירוע בו הטופס נסגר. |  |

# ביבליוגרפיה

האתר הרשמי להורדת python 2.7:

<https://www.python.org/>

הורדת pip:

[click to download pip](https://www.google.co.il/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwiii8zmg87MAhWlK8AKHfTCAX4QFggoMAE&url=https%3A%2F%2Fbootstrap.pypa.io%2Fget-pip.py&usg=AFQjCNE8Fo9j_sgo1hBzEoUT39H85hFDrg&cad=rja)

פורום עזרה למתכנתים:

<http://stackoverflow.com/>

התקנת pyCrypto על פי ההוראות כאן:  
<https://pypi.python.org/pypi/pycrypto>  
התקנת pprp על פי ההוראות כאן:  
<https://pypi.python.org/pypi/pprp/0.2.5>  
מדריכים לפעולות ומחלקות בc#:  
[https://msdn.microsoft.com/](https://msdn.microsoft.com/en-us/library/aa287558(v=vs.71).aspx)   
מימוש יעיל של אלגוריתם דיפי-הלמן בC#:  
<https://www.codeproject.com/Articles/24632/Shared-Key-Generation-using-Diffie-Hellman>  
השוואה בין פונקציות hash שונות:  
<http://automationrhapsody.com/md5-sha-1-sha-256-sha-512-speed-performance>   
פונקציות להצפנה ופענוח סימטריים:  
<http://stackoverflow.com/a/10177020/5631525>

# קוד צד השרת-

**Server.py  
  
import** socket  
**import** sys  
**import** threading **as** th  
**import** hashlib  
**import** os.path  
**import** xml.etree.ElementTree **as** ET  
**import** xml.dom.minidom  
**import** select  
**import** pprp  
**import** pprp.config  
**import** io  
**from** Crypto.Cipher **import** AES  
**import** binascii  
**import** math  
**from** Crypto.Cipher **import** PKCS1\_OAEP  
  
registered\_profiles = []  
openThreads = []  
openAuthenticatedClients = {}  
openClients = []  
Messages\_to\_send = []  
  
  
*#<editor-fold desc= "DiffieHellman class">***"""  
This class is an implementation of the diffie hellman algorithm in python  
it does not create the variables on its own but is always the respond side of  
the algorithm.  
"""  
class** DiffieHellman:  
 *""" Class to represent the Diffie-Hellman key exchange protocol """  
 # Current minimum recommendation is 2048 bit. p|g|(g^a)%p* **def** \_\_init\_\_(self, new\_p, new\_g):  
 self.p = new\_p  
 self.g = new\_g  
 self.\_\_a = int(binascii.hexlify(os.urandom(32)), base=16)  
  
 **def** get\_private\_key(self):  
 *""" Return the private key (a) """* **return** self.\_\_a  
  
 **def** gen\_public\_key(self):  
 *""" Return A, A = g ^ a mod p """  
 # calculate G^a mod p* **return** pow(long(self.g), self.\_\_a, long(self.p))  
  
 **def** gen\_shared\_key(self, other\_contribution):  
 *""" Return g ^ ab mod p """  
 # calculate the shared key G^ab mod p* shared\_key = pow(long(other\_contribution), self.\_\_a, long(self.p))  
 **return** str(shared\_key)  
  
*#</editor-fold>  
  
  
# A receiving thread, receive messages from one client after the session has begun.***class** ReceivingThread(th.Thread):  
 **def** \_\_init\_\_(self, client):  
 th.Thread.\_\_init\_\_(self)  
 self.client = client  
  
 **def** run(self):  
 **while** True:  
 data = self.client.recv(4096)  
 **if "Exit" in** data:  
 self.client.close()  
 **break** Messages\_to\_send.append((data, self.client))  
 set\_connection\_status(self.client, False)  
  
  
*# A sending thread, send messages to a client after the session has begun.***class** SendingThread(th.Thread):  
 **def** \_\_init\_\_(self, client):  
 th.Thread.\_\_init\_\_(self)  
 self.client = client  
  
 **def** run(self):  
 **while** True:  
 **for** (message, sending\_client) **in** Messages\_to\_send:  
 **if** sending\_client **is not** self.client:  
 self.client.send(message)  
 Messages\_to\_send.remove((message, sending\_client))  
 **else**:  
 **continue** set\_connection\_status(client, False)  
  
  
*# A profile of a client. contains the socket connected to him, username,  
# whether he is connected or not and the encryption key shared with him.***class** Profile:  
 **def** \_\_init\_\_(self, client, gaxy):  
 self.socket = client  
 self.username = **""** self.is\_connected = False  
 self.key = gaxy  
  
 **def** get\_socket(self):  
 **return** self.socket  
  
 **def** get\_key(self):  
 **return** self.key  
  
 **def** get\_username(self):  
 **return** self.username  
  
 **def** is\_username\_set(self):  
 **if** username == **""**:  
 **return** False  
 **return** True  
  
 **def** set\_username(self, username):  
 self.username = username  
  
 **def** get\_connection\_status(self):  
 **return** self.is\_connected  
  
 **def** set\_connection\_status(self, boolean):  
 self.is\_connected = boolean  
  
  
*# A client after the session has begun.***class** Client:  
 **def** \_\_init\_\_(self, ReceivingThread, SendingThread):  
 self.receive = ReceivingThread  
 self.send = SendingThread  
 self.isclosed = False  
  
 **def** close(self):  
 self.receive.join()  
 self.send.join()  
 self.isclosed = True  
  
 **def** \_\_run\_\_(self):  
 self.receive.start()  
 self.send.start()  
  
 **def** get\_isclosed(self):  
 **return** self.isclosed  
  
  
*# A session, contains two clients.***class** SessionThread(th.Thread):  
 **def** \_\_init\_\_(self, client1, client2):  
 th.Thread.\_\_init\_\_(self)  
 self.client1 = client1  
 client1.\_\_run\_\_()  
 self.client2 = client2  
 client2.\_\_run\_\_()  
 self.isclosed = False  
  
 **def** run(self):  
 **while** (**not** client1.get\_isclosed()) **and** (**not** client2.get\_isclosed()):  
 **continue** set\_connection\_status(client1, False)  
 set\_connection\_status(client2, False)  
 client1.close()  
 client2.close()  
 self.isclosed = True  
  
 **def** get\_is\_closed(self):  
 **return** self.isclosed  
  
*# Xor encryption or decryption method, receive key and input and returns the xor result.***def** xor\_encrypt\_decrypt(key, input2): *# Can be any chars, and any size array* output = []  
 **for** i **in** range(len(input2)):  
 xor\_num = ord(input2[i]) ^ ord(key[i % len(key)])  
 output.append(chr(xor\_num))  
  
 return\_str = str(**''**.join(output))  
 return\_str = return\_str[0:return\_str.\_\_len\_\_()-1]  
 **return** return\_str  
  
  
*# Set the profile's username according to his socket.***def** set\_profile\_username(client, username):  
 **for** profile **in** registered\_profiles:  
 **if**(profile.get\_socket() **is** client):  
 profile.set\_username(username)  
  
  
*# Set the profile's connection status according to his socket.***def** set\_connection\_status(client, boolean):  
 **for** profile **in** registered\_profiles:  
 **if** profile.get\_socket() **is** client :  
 profile.set\_connection\_status(boolean)  
  
  
*# Get a profile's connection status according to his username.***def** is\_user\_loggedin(username):  
 **for** profile **in** registered\_profiles:  
 **if** profile.get\_username() == username:  
 **return** profile.get\_connection\_status()  
  
  
*# Get a profile's encryption key according to his socket.***def** get\_profile\_key\_by\_socket(socket):  
 **for** profile **in** registered\_profiles:  
 **if** profile.get\_socket() == socket:  
 **return** profile.get\_key()  
  
  
*# Get a profile's encryption key according to his username.***def** get\_profile\_key\_by\_username(username):  
 **for** profile **in** registered\_profiles:  
 **if** profile.get\_username() == username :  
 **return** profile.get\_key()  
  
  
*# Get a profile's socket according to his username.***def** get\_profile\_socket(username):  
 **for** profile **in** registered\_profiles:  
 **if** profile.get\_username() == username :  
 **return** profile.get\_socket()  
  
  
*# Get a profile's username according to his socket.***def** get\_profile\_username(client):  
 **for** profile **in** registered\_profiles:  
 **if** profile.get\_socket() **is** client :  
 **return** profile.get\_username()  
  
  
*# Adds the data length (4 bytes) to the beginning of a message and sends to the client.***def** send\_message(client, message):  
 client.send(get\_data\_length(message.\_\_len\_\_()) + message)  
  
  
*# Gets the data length represented in 4 bytes string according to a given integer***def** get\_data\_length(data\_length):  
 s = **""  
 if** data\_length < 10:  
 s = **"000"** + str(data\_length)  
 **elif** data\_length < 100:  
 s = **"00"** + str(data\_length)  
 **elif** data\_length < 1000:  
 s = **"0"** + str(data\_length)  
 **else**:  
 s = str(data\_length)  
 **return** s  
  
**global** root\_node  
**global** xmlfile  
  
  
**def** main():  
 reload(sys)  
 sys.setdefaultencoding(**'latin-1'**)  
 xmlfile = open(**"resource.txt"**, **'w'**)  
 **try**:  
 xmlfiler = open(**"resource.txt"**, **'r'**)  
 root\_node = ET.fromstring(xmlfiler.read())  
  
 **except**:  
 root\_node = ET.Element(**"root"**)  
 s\_socket = socket.socket()  
 s\_socket.bind((**"0.0.0.0"**, 11000))  
 s\_socket.listen(5)  
 doc = ET.SubElement(root\_node, **"doc"**)  
  
 **while** True:  
 r, w, x = select.select([s\_socket] + openClients, openClients, [])  
 h = hashlib.sha256()  
 curr\_active\_clients = **""  
 for** Thread **in** openThreads:  
 **if**(Thread.get\_is\_closed()):  
 OpenThreads.remove(Thread)  
  
 **for** c\_socket **in** r:  
 **if** c\_socket **is** s\_socket: *# First arrive.* message = []  
 (new\_socket, address) = s\_socket.accept()  
 openClients.append(new\_socket)  
 data = new\_socket.recv(1024) *# diffiehellman request* message = data.split(**"|"**)  
 server\_dif = DiffieHellman(message[0], message[1])  
 shared\_key = str(server\_dif.gen\_shared\_key(message[2]))  
 registered\_profiles.append(Profile(new\_socket, shared\_key))  
 new\_socket.send(str(server\_dif.gen\_public\_key()))  
  
 **else**:  
 message = **" "** data = c\_socket.recv(1024)  
 length = int(data[0:4])  
 message = data[0:length+4]  
  
 **if "/signin" in** message:  
 username = **""** password\_enc = **""** z = 0  
 **for** i **in** message[11:]:  
 **if** i == **'@'**:  
 z += 1  
 **break** username += i  
 z += 1  
 **for** j **in** message[11 + z:]:  
 password\_enc += j  
 password\_plain = xor\_encrypt\_decrypt(get\_profile\_key\_by\_socket(c\_socket), password\_enc) *#decrypt the password.* **if not** (doc.find(username) **is** None): *# User exist in the system.* called\_member = doc.find(username)  
 h.update(password\_plain) *# Sign in case.* **if** h.digest() == called\_member.text: *# User exist exactly in the system.* **if** is\_user\_loggedin(username): *# User is already logged in.* response = **"EntryForm Error2:AuthentificationFailed"** send\_message(c\_socket, response)  
 **else**: *# Authentification succeeded.* response = **"EntryForm AuthenticationSucceeded "** send\_message(c\_socket, response)  
 set\_profile\_username(c\_socket,username)  
 set\_connection\_status(c\_socket, True)  
 **else**: *# Password is wrong.* response = **"EntryForm Error2:AuthentificationFailed"** send\_message(c\_socket, response)  
 **else**: *# There is no such user in the system.* response = **"EntryForm Error2:AuthentificationFailed"** send\_message(c\_socket, response)  
  
 **elif "/signup" in** message:  
  
 username = **""** password\_enc = **""** z = 0  
 **for** i **in** message[11:]:  
 **if** i == **'@'**:  
 z += 1  
 **break** username += i  
 z += 1  
 **for** j **in** message[11 + z:]:  
 password\_enc += j  
  
 password\_plain = xor\_encrypt\_decrypt(get\_profile\_key\_by\_socket(c\_socket), password\_enc)  
 **print** password\_plain  
 **if** doc.find(username) **is** None: *# Sign up case.* h.update(password\_plain)  
 ET.SubElement(doc, username).text = h.digest()  
 h = hashlib.sha256()  
 response = **"EntryForm SignedUpSuccessfully"** send\_message(c\_socket, response)  
 ET.ElementTree(root\_node).write(xmlfile)  
 **else**:  
 response = **"SignUpForm Error1:UsernameTaken"** send\_message(c\_socket, response)  
  
 **elif "startsession" in** message:  
 called\_member = **""** username2=**""  
 for** i **in** message[17:]:  
 **if** i == **' '**:  
 **break** called\_member += i  
 **if** is\_user\_loggedin(called\_member):  
  
 called\_socket = get\_profile\_socket(called\_member)  
 username2 = get\_profile\_username(c\_socket)  
 response = **"SessionForm startsession "** + username2  
 send\_message(called\_socket, response)  
 **else**:  
 **print** is\_user\_loggedin(called\_member)  
 response = **"SessionForm Error3RequestedUserIsNotOnlineAnymore"** send\_message(c\_socket, response)  
  
 **elif "acceptsession" in** message: *# after this will come the name of the session starter.* called\_member = **""  
  
 for** i **in** message[18:]:  
 **if** i == **' '**:  
 **break** called\_member += i  
 username = get\_profile\_username(c\_socket)  
  
 **if** is\_user\_loggedin(called\_member):  
 client = get\_profile\_socket(called\_member)  
 openClients.remove(client)  
 openClients.remove(c\_socket)  
 openThreads.append(SessionThread(Client(ReceivingThread(client), SendingThread(client)), Client(ReceivingThread(c\_socket), SendingThread(c\_socket))))  
 response = **"SessionForm acceptsession "** send\_message(client=get\_profile\_socket(username), message=response + **" "** + called\_member)  
 **print** username *# For example, "0040SessionForm acceptsession AVIV"* set\_connection\_status(get\_profile\_socket(username), False)  
  
 send\_message(client=get\_profile\_socket(called\_member), message=response + **" "** + username)  
 **print** called\_member  
 set\_connection\_status(get\_profile\_socket(called\_member), False)  
  
 **else**:  
 response = **"SessionForm Error3RequestedUserIsNotOnlineAnymore"** send\_message(client=c\_socket, message=response)  
  
 **elif** message[4:15] == **"refreshList"**:  
  
 **for** profile **in** registered\_profiles:  
 **if not**(profile.get\_socket() **is** c\_socket):  
 **if**( profile.get\_connection\_status()):  
 curr\_active\_clients += profile.get\_username()  
 curr\_active\_clients += **"/"** response = **"SessionForm refreshList "** + curr\_active\_clients  
 send\_message(client=c\_socket, message=response)  
  
 **elif "denyCall" in** message:  
 username = **""  
  
 for** i **in** message[13:]:  
 **if** i == **' '**:  
 **break** username += i  
  
 **if** is\_user\_loggedin(str(username)):  
 response = **"SessionForm denied "** + str(username)  
 send\_message(client=c\_socket, message=response)  
  
 **elif "EXIT" in** message:  
  
 openClients.remove(c\_socket)  
 set\_connection\_status(c\_socket, False)  
 c\_socket.close()  
  
 **else**:  
  
 **print "Message didn't fall into any of the cases: "**+ message  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()  
 xmlfile.write(ET.tostring(root\_node))  
 xmlfile.close()

# קוד צד הלקוח-

**Communicator.cs:**using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.IO;

using System.Threading;

using System.Windows.Forms;

using System.Security.Cryptography;

using System.Numerics;

namespace WindowsFormsApplication1

{

public class Communicator

{

/// <summary>

/// Singleton- instance of this class.

/// </summary>

private static Communicator instance;

/// <summary>

/// The thread that will provide the messages from the server.

/// </summary>

private static Thread RecievingThread;

/// <summary>

/// The socket connection to the server.

/// </summary>

private static Socket srvr;

/// <summary>

/// A list that contains all current open forms.

/// </summary>

private static List<baseForm> openForms;

/// <summary>

/// Boolean variable - True: talking to server, False: talking to client.

/// </summary>

static bool talkingTo = true;

/// <summary>

/// Access control.

/// </summary>

static int i = 0;

/// <summary>

/// Encryption key shared with the other client.

/// </summary>

static string addresseeKey;

/// <summary>

/// Encryption key shared with server.

/// </summary>

static string serverKey;

/// <summary>

/// DiffieHellman algorithm with client.

/// </summary>

static DiffieHellman clientDH;

/// <summary>

/// DiffieHellman algorithm with server.

/// </summary>

static DiffieHellman serverDH;

/// <summary>

/// static constructor, singleton static class structure.

/// </summary>

static Communicator( ) { }

/// <summary>

/// Singleton static class structure.

/// </summary>

public static Communicator Instance

{

get

{

if (instance == null)

{

instance = new Communicator();

openForms = new List<baseForm>();

}

return instance;

}

}

/// <summary>

/// Combine all byte arrays from an array of them.

/// </summary>

/// <param name="arrays"> An array of byte arrays.</param>

/// <returns> A byte arrays contains all given.</returns>

internal static byte[] CombineByteArrays(byte[][] arrays)

{

return arrays.SelectMany(x => x).ToArray();

}

/// <summary>

/// Set the given data length to 4-length byte array.

/// </summary>

/// <param name="dataLength"> integer represent the length of data</param>

/// <returns> 4 byte length byte array</returns>

internal static byte[] GetDataLength(int dataLength)

{

string s;

if (dataLength < 10)

s = "000" + dataLength.ToString();

else

if (dataLength < 100)

s = "00" + dataLength.ToString();

else

if (dataLength < 1000)

s = "0" + dataLength.ToString();

else

s = dataLength.ToString();

return Encoding.ASCII.GetBytes(s);

}

/// <summary>

/// Set the client's shared key and get the response to be sent.

/// </summary>

/// <param name="request"> The diffie-hellman request message.</param>

/// <returns> DiffieHellman object represent response.</returns>

public static DiffieHellman GetResponseMessage(string request)

{

if (talkingTo)

{

clientDH = new DiffieHellman(256).GenerateResponse(request);

addresseeKey = Convert.ToBase64String(clientDH.Key);

Console.WriteLine("Addressee KEY: " + addresseeKey);

return clientDH;

}

else

return null;

}

/// <summary>

/// Get the diffie-hellman request message.

/// </summary>

/// <returns> DiffieHellman object represents request.</returns>

public static DiffieHellman GetRequestMessage()

{

if (talkingTo)

{

clientDH = new DiffieHellman(256).GenerateRequest();

return clientDH;

}

else

return null;

}

/// <summary>

/// Set the client's shared key.

/// </summary>

/// <param name="response"> The diffie-hellman response message. </param>

public static void HandleResponse(string response)

{

if (talkingTo)

{

addresseeKey = Convert.ToBase64String(clientDH.HandleResponse2(response));

Console.WriteLine("Addressee KEY: "+addresseeKey);

}

}

/// <summary>

/// Sends the message after encryption to the other client or without encryption to the server.

/// </summary>

/// <param name="sender"> The sending form name.</param>

/// <param name="message"> The message to be sent.</param>

/// Used mostly to the other client.

public static void SendMessage(string sender, string message)

{

if (srvr != null)

{

i++;

byte[] data = new byte[4096];

string data2 = sender + " " + message + " "; /// The plain data to be encrypted.

if (talkingTo) /// Talking to server - non encrypted messages.

{

byte[] lengthOfData = new byte[4];

/// 4 byte data length.

lengthOfData = GetDataLength(data2.Length);

/// Sends the data and additional first 4 bytes represent the length.

srvr.Send(CombineByteArrays(new byte[][] { lengthOfData, Encoding.ASCII.GetBytes(data2) }));

}

else ///Talking to client - encrypted messages.

{

/// Encrypted data.

byte[] datatosend = Encoding.ASCII.GetBytes(Cryptography.Encrypt(data2, addresseeKey));

///Sends the data and additional first 4 bytes represent the length.

srvr.Send(CombineByteArrays(new byte[][] { GetDataLength(datatosend.Length), datatosend }));

}

}

}

/// <summary>

/// Sends the message after encryption to the other client or without encryption to the server.

/// </summary>

/// <param name="message">The message to be sent.</param>

/// Used mostly to the server.

public static void SendMessage(string message)

{

if (srvr != null)

{

i++;

byte[] data = new byte[4096];

string data2 = message + " "; ///plain data to send.

Console.WriteLine(data2);

if (talkingTo) //Talking to server- non encrypted messages.

{

byte[] lengthOfData = new byte[4];

/// 4 byte data length.

lengthOfData = GetDataLength(data2.Length);

/// Sends the data and additional first 4 bytes represent the length.

srvr.Send(CombineByteArrays(new byte[][] { lengthOfData, Encoding.ASCII.GetBytes(data2) }));

}

else ///Talking to client - encrypted messages.

{

/// Encrypted data.

byte[] datatosend = Encoding.ASCII.GetBytes(Cryptography.Encrypt(data2, addresseeKey));

///Sends the data and additional first 4 bytes represent the length.

srvr.Send(CombineByteArrays(new byte[][] { GetDataLength(datatosend.Length), datatosend }));

}

}

}

/// <summary>

/// Add a new form to the forms' list.

/// </summary>

/// <param name="newForm"> the new form.</param>

public static void AddForm(baseForm newForm)

{

if (openForms != null)

{

openForms.Add(newForm);

}

}

/// <summary>

/// Removes the specified form from the forms' list.

/// </summary>

/// <param name="removedForm"></param>

public static void RemoveForm(baseForm removedForm)

{

if (openForms != null)

{

if (openForms.Contains(removedForm))

{

openForms.Remove(removedForm);

}

}

}

/// <summary>

/// Recieve data and invoke the data to the specified form in it.

/// </summary>

public static void RecievingMessage()

{

if (srvr != null && openForms != null)

{

bool sent = true; // boolean represent whether the data was sent.

int i = 0; //access control

int length = 0; // access control

string msg; // access control

string[] msg\_array; //access control

if (srvr != null) ///There is a connection to the server.

while (true)

{

Thread.Sleep(20);

byte[] fullData = new byte[4096];

///Recieve the data here.

int bytesRec = srvr.Receive(fullData);

///Gets the string represent the data.

string stringData = Encoding.ASCII.GetString(fullData);

///Gets the data length from the first 4 bytes.

length = int.Parse(string.Join("",stringData.Take(4).ToArray()));

///Sent set to false.

sent = false;

if (talkingTo) //Talking to server- non encrypted messages.

{

/// Takes only the length of data from the first 4 bytes.

byte[] data = fullData.Skip(4).Take(length).ToArray();

/// Gets the message.

msg = Encoding.ASCII.GetString(data);

}

else // Talking to client- encrypted messages.

{

/// Taking only the encrypted data - without the bytes represent length.

stringData = string.Join("", stringData.Skip(4).Take(length).ToArray());

/// Decrypts the data.

msg = Cryptography.Decrypt(stringData, addresseeKey);

}

///Split the data according to " " seperator.

msg\_array = msg.Split(' ');

///Checking to which form the message is destined.

for (i = 0; i < openForms.Count; i++)

{

///If the form name equals to a form in the list.

if (msg\_array[0].Equals(openForms[i].Name))

{

///Invoke the Recieve method at the specified form.

openForms[i].Recieve(msg\_array);

///Set sent to true.

sent = true;

}

}

///If message was not sent.s

if (!sent)

Console.WriteLine("Default case, Communicator " + msg\_array[0]);

}

}

}

/// <summary>

/// Encrypt or decrypt using XOR.

/// </summary>

/// <param name="input"> The given data to encrypt/decrypt./</param>

/// <returns> The data after encryption/decryption. </returns>

public static string xorEncryptDecrypt(string input)

{

Console.WriteLine("ServerKey: "+serverKey);

Console.WriteLine("Input: "+ input);

// check for null, empty strings etc.

if (String.IsNullOrEmpty(input) || String.IsNullOrEmpty(serverKey))

return input;

// Do not append string in the loop; use StringBuilder for this

StringBuilder cyphertext = new StringBuilder(input.Length);

// You have no need in SubString and other stuff: just char xor char

// i % pad.Length - if pad is shorter than plaintext

for (int i = 0; i < input.Length; ++i)

cyphertext.Append((char)(input[i] ^ serverKey[i % serverKey.Length]));

Console.WriteLine(cyphertext.ToString());

return cyphertext.ToString();

}

/// <summary>

/// Returns true if the client is connected to the server. false otherwise.

/// </summary>

/// <returns> boolean variable represents whether the connection was or wasn't established.</returns>

public static bool IsConnectedToServer()

{

return srvr.Connected;

}

/// <summary>

/// Set that the client is now talking to the client and not to the server anymore.

/// </summary>

public static void SetToClient()

{

talkingTo = false;

}

/// <summary>

/// Establish the connection to the server.

/// </summary>

/// <param name="ipAddress">IP address of the server.</param>

/// <param name="port"> Port to which the server is listening.</param>

public static void Connect(IPAddress ipAddress, int port)

{

openForms = new List<baseForm>();

talkingTo = true;

IPEndPoint remoteEP = new IPEndPoint(ipAddress, 11000);

try

{

byte[] data = new byte[4096];

srvr = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

srvr.Connect(remoteEP); //Connects to server

serverDH = new DiffieHellman(256).GenerateRequest(10);

srvr.Send(Encoding.ASCII.GetBytes(serverDH.ToString()));

int bytesrec= srvr.Receive(data);

serverDH.HandleResponse(Encoding.ASCII.GetString(data));

serverKey = new BigInteger(serverDH.Key).ToString();

Console.WriteLine("Connected to server.");

RecievingThread = new Thread(new ThreadStart((RecievingMessage)));

RecievingThread.IsBackground = true;

RecievingThread.Start();

Console.WriteLine("Thread started.");

}

catch(ThreadStartException e)

{

Console.WriteLine ("{0} Exception caught.", e);

}

catch (ArgumentNullException ane)

{

Console.WriteLine("{0} Exception caught.", ane);

}

catch (SocketException se)

{

Console.WriteLine("{0} Exception caught.", se);

}

catch (Exception ex)

{

Console.WriteLine("{0} Exception caught.", ex);

}

}

/// <summary>

/// Disconnects from the server safely.

/// </summary>

public static void DisConnect()

{

if(!talkingTo)

srvr.Send(Encoding.ASCII.GetBytes("Exit"));

RecievingThread.Abort();

srvr.Disconnect(false);

}

}

}

**firstForm.cs:**using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net.Sockets;

using System.Net;

namespace WindowsFormsApplication1

{

/// <summary>

/// Connect to the server here.

/// </summary>

/// The first form of the project. This form will open and its main target is to establish

/// the connection between the client and the server.

public partial class firstForm : Form

{

/// <summary>

/// Creates the form.

/// </summary>

public firstForm()

{

InitializeComponent();

label6.Text = "Enter the server's IP and port." + Environment.NewLine + "then press connect:";

// label 6 is the text above the two text boxes.

IP\_textBox.Text = "127.0.0.1";

Port\_textBox.Text = "11000";

}

EntryForm lf; /// an instance of the next form.

/// <summary>

/// This event will run once the form is loaded.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

///Setting the next form's instance.

lf = new EntryForm();

}

/// <summary>

/// This event will run once the client press the "connect" button.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

string[] ip\_array = IP\_textBox.Text.Split('.');

int port = 0;

if (ip\_array.Length == 4 && int.TryParse(Port\_textBox.Text, out port)) ///Check the client's input.

{

if (port > 1024 && port < 65535) ///Check whether the port is valid integer.

{

/// trying to connect to the server.

Communicator.Connect(new IPAddress(new byte[] { Convert.ToByte(int.Parse(ip\_array[0])), Convert.ToByte(int.Parse(ip\_array[1])), Convert.ToByte(int.Parse(ip\_array[2])), Convert.ToByte(int.Parse(ip\_array[3])) }), int.Parse(Port\_textBox.Text));

/// if the connection succeeded.

if (Communicator.IsConnectedToServer())

{

MessageBox.Show("You have connected successfully to the server. \r\n Press OK to continue.");

this.BeginInvoke((Action)lf.Show);

/// Hides because otherwise the main thread of the program will be shut down.

this.Hide();

}

else

/// Connection to server failed.

MessageBox.Show("Server is either closed or not available. \r\n Please try again later.");

}

else

MessageBox.Show("You've entered an incorrect port.");

}

else

MessageBox.Show("You've entered incorrect port or IP address.");

}

}

}

**loginForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net.Sockets;

using System.Net;

namespace WindowsFormsApplication1

{

/// <summary>

/// Log in the server's system here.

/// </summary>

/// This form is the form where the client logs in the system.

/// The connection to the server is already established.

public partial class EntryForm : baseForm

{

SignUpForm signUpForm; // Sign up form variable.

SessionForm sf; // Session form variable.

bool ClosedByClient; // Boolean variable that represents whether the form was closed by client or

/// <summary>

/// Creates the form.

/// </summary>

public EntryForm()

{

InitializeComponent();

}

/// <summary>

/// This event will run once the form is loaded.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Communicator.AddForm(this);

sf = new SessionForm(); // Session form variable.

signUpForm = new SignUpForm(this); // Sign up form variable.

ClosedByClient = true; // Setting the boolean to default of true.

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array == null) // Just in case.

{

}

else

{

switch (msg\_array[1]) //Suppose to contain one of the next cases.

{

case "Error2:AuthentificationFailed": //The password or username were wrong.

MessageBox.Show("Either your password or your username was wrong, please try again to enter the system."); break;

case "AuthenticationSucceeded": //The password and username were right.

{

MessageBox.Show("Authentification succeeded.");

Communicator.RemoveForm(this);

ClosedByClient = false;

this.BeginInvoke(new Action(sf.Show));

this.Hide();

ClosedByClient = true;

}

break;

case "SignedUpSuccessfully": // The client managed to sign up.

{

MessageBox.Show("Signed up successfully, you may now sign in the system.");

Communicator.RemoveForm(signUpForm); // removes the signup form from the communicator.

if (signUpForm.InvokeRequired)

signUpForm.Invoke(new Action(signUpForm.Close)); //Closes the sign up form.

else

signUpForm.Close(); //Closes the sign up form.

signUpForm = new SignUpForm(this);// setting a new instance of signupform

SignUpButton.Enabled = true;

}

break;

default: //default input case.

Console.WriteLine("Default case, " + this.Name + " " + msg\_array[1]);

break;

}

}

}

/// <summary>

/// Shows the sign up form whenever the sign up button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

SignUpButton.Enabled = false;

signUpForm.Show(); //Shows the sign up form.

Communicator.AddForm(signUpForm); //adds it to the communicator's list.

}

/// <summary>

/// Tries to sign in to the system whenever the sign in button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

Communicator.SendMessage("/signin" + this.UsernameTextBox.Text + "@" + Communicator.xorEncryptDecrypt( this.PasswordTextBox.Text));

}

/// <summary>

/// Removes the signup form from the communicator and setting a new instance.

/// </summary>

public void SignUpForm\_closed()

{

Communicator.RemoveForm(signUpForm);

SignUpButton.Enabled = true;

signUpForm = new SignUpForm(this);

}

/// <summary>

/// Removes the form from communicator and notify the server whenever this form is being closed by the client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void LoginForm\_FormClosed(object sender, FormClosedEventArgs e)

{

if (ClosedByClient)

{

ClosedByClient = false;

Communicator.SendMessage("EXIT"); //notify the server about exit.

Communicator.RemoveForm(this); //removes form from communicator.

Communicator.DisConnect(); //disconnects from server.

this.Close();

}

}

private void label1\_Click(object sender, EventArgs e)

{

}

}

}

**SignUpForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

/// <summary>

/// A form where the client is able to sign up into the system

/// and create his own profile.

/// </summary>

public partial class SignUpForm : baseForm

{

private EntryForm lf;

/// <summary>

/// Creates the form.

/// </summary>

public SignUpForm(EntryForm lf)

{

InitializeComponent();

this.lf = lf;

}

/// <summary>

/// Checks the client's input and tries to sign up whenever the signup button is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (this.UsernameTextBox.Text == "")

MessageBox.Show("You didn't enter a username. please try again.");

else

{

if (this.PasswordTextBox.Text == "" || this.PasswordTextBox.TextLength < 5 || this.PasswordTextBox.TextLength > 10)

{

MessageBox.Show("You didn't enter an appropriate password. please try again." + Environment.NewLine + "Make sure your password is larger than 5 characters and smaller than 10.");

}

else

{

Communicator.SendMessage("/signup" + this.UsernameTextBox.Text + "@" + Communicator.xorEncryptDecrypt(this.PasswordTextBox.Text));

}

}

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array == null)

{

}

else

{

switch (msg\_array[1])

{

case "Error1:UsernameTaken":

MessageBox.Show("Your chosen username is already taken, please choose another one and try again."); break;

}

}

}

/// <summary>

/// Invoke the login form's "SignUpForm\_closed" method whenever this form is closing.

/// </summary>

/// <param name="sender">event parameter.</param>

/// <param name="e">event parameter.</param>

private void SignUpForm\_FormClosing(object sender, FormClosingEventArgs e)

{

lf.Invoke((Action)lf.SignUpForm\_closed);

}

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

{

}

}

}

**SessionForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Numerics;

using System.Threading;

namespace WindowsFormsApplication1

{

/// <summary>

/// Call another client here.

/// </summary>

/// In this form the client will search for another client connected

/// to the server and will call him.

/// The client is called to this form, here he can answer or ignore a call.

public partial class SessionForm : baseForm

{

bool isPickedAUser; //access control

bool isBeingCalled; //access control

bool isCalled; //access control

string userCalled; //access control

/// <summary>

/// Creates the form.

/// </summary>

public SessionForm()

{

InitializeComponent();

}

/// <summary>

/// Set the access controls whenever this form is loaded.

/// </summary>

/// <param name="sender"> event parameter. </param>

/// <param name="e"> event parameter. </param>

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

{

ignoreCallButton.Enabled = false;

answerCallButton.Enabled = false;

isBeingCalled = false;

isCalled = false;

isPickedAUser = false;

usersList.MultiSelect = false;

Communicator.AddForm(this);

}

/// <summary>

/// Set the boolean variable to true whenever the client picked another client from the list.

/// </summary>

/// <param name="sender"> event parameter. </param>

/// <param name="e"> event parameter. </param>

private void listView1\_SelectedIndexChanged(object sender, EventArgs e)

{

isPickedAUser = true;

}

/// <summary>

/// Asks the server for a new list of clients whenever the refresh button is clicked.

/// </summary>

/// <param name="sender">event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

Communicator.SendMessage("refreshList");

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array == null)

{

}

else

{

switch (msg\_array[1])

{

case "refreshList":

{

string[] listUsers;

listUsers = msg\_array[2].Split('/'); //split the string by '/'.

usersList.Items.Clear();

foreach (string userName in listUsers)

{

usersList.Items.Add(userName); // adds the Items to the list.

}

}

break;

case "startsession":

{

isBeingCalled = true;

MessageBox.Show("You are called by " + msg\_array[2] +"\r\n press ok and answer or ignore the call.");

answerCallButton.Enabled = true;

ignoreCallButton.Enabled = true;

userCalled = msg\_array[2];

}

break;

case "denied":

{

MessageBox.Show("Your call got denied.");

isCalled = false;

}

break;

case "acceptsession":

{

mainForm mF = new mainForm();

Communicator.AddForm(mF);

if (isCalled)

{

Thread.Sleep(20);

Communicator.SendMessage(mF.Name, "/RequestDiffie " + Communicator.GetRequestMessage().ToString());

}

this.Hide();

this.BeginInvoke((Action)mF.Show);

isCalled = false;

}

break;

default:

{

MessageBox.Show(msg\_array[1]);

}

break;

}

}

}

/// <summary>

/// Calling the specified user if picked whenever the call button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

if (isPickedAUser)

{

Communicator.SendMessage("startsession " + usersList.SelectedItems[0].Text);

isCalled = true;

}

else

MessageBox.Show("No user was chosen to be called.");

}

/// <summary>

/// Ignores the call whenever the ignore button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

if(isBeingCalled)

{

Communicator.SendMessage("denyCall " + userCalled);

}

else ///Logically, this is not suppose to happen.

{

MessageBox.Show("Sorry, it was suppose to be hidden. Thanks for your awarness!");

}

ignoreCallButton.Enabled = false;

answerCallButton.Enabled = false;

}

/// <summary>

/// Answer the call whenever the answer call is being clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (isBeingCalled)

{

Communicator.SendMessage("acceptsession " + userCalled +" ");

}

else //Logically, this is not suppose to happen.

{

MessageBox.Show("Sorry, it was suppose to be hidden. Thanks for your awarness!");

}

answerCallButton.Enabled = false;

ignoreCallButton.Enabled = false;

}

/// <summary>

/// Removes the form from communicator and notify the server whenever this form is being closed by the client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void SessionForm\_FormClosed(object sender, FormClosedEventArgs e)

{

Communicator.SendMessage("EXIT");

Communicator.RemoveForm(this);

Thread.Sleep(20);

Communicator.DisConnect();

this.Close();

}

}

}

**mainForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net;

using System.Net.Sockets;

using System.IO;

using System.Threading;

using System.Numerics;

namespace WindowsFormsApplication1

{

public enum TicValue { E,X,O };

/// <summary>

/// This form is shared between the connected clients.

/// </summary>

public partial class mainForm : baseForm

{

/// <summary>

/// Creates the form.

/// </summary>

public mainForm()

{

InitializeComponent();

}

Bitmap cursor; // access controls

PictureBox pictureBox;// access controls

Notepad Np; // Notepad form

TicTacToe tic;// TicTacToe Form

Paint Pf; // Paint form

int i = 0;// access controls

bool flag= false;// access controls

/// <summary>

/// This event will run once the form is loaded.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Communicator.AddForm(this);// Add the form to the communicator's list.

cursor = Properties.Resources.Cursor;

pictureBox = new PictureBox(); //Set picture.

pictureBox.Image = cursor; //Set picture.

pictureBox.Size = new Size(34, 37); //Set picture's size.

Controls.Add(pictureBox); //Adds the picture.

Np = new Notepad();

Pf = new Paint();

this.NotepadButton.Enabled = false;

this.TiTaToButton.Enabled = false;

MessageBox.Show("Wait until connection is established. \r\n Buttons will be unavailable till then.");

}

/// <summary>

/// Whenever the mouse moves this method update the mouse of the other client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void mainForm\_MouseMove(object sender, MouseEventArgs e)

{

if (this.NotepadButton.Enabled && this.TiTaToButton.Enabled && this.PaintButton.Enabled) //while only main form is open.

{

if (i % 5 == 0) //Efficiency.

{

Communicator.SendMessage(this.Name, ("/Mouse\_pos " + (Cursor.Position.X - this.Left) + " " + (Cursor.Position.Y - this.Top+150) + " "));

}

i++;

}

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array == null)

{

}

else

{

try

{

switch (msg\_array[1])

{

case "/Window\_size":

{

flag = false;

this.Height = int.Parse(msg\_array[2]); //Set the new window size.

this.Width = int.Parse(msg\_array[3]); //Set the new window size.

}

break;

case "/RequestDiffie":

{

Communicator.SendMessage(this.Name, "/FinalDiffie " + Communicator.GetResponseMessage(msg\_array[2]));

Communicator.SetToClient();

this.NotepadButton.Enabled = true;

this.TiTaToButton.Enabled = true;

} break;

case "/FinalDiffie":

{

Console.WriteLine("RECEIVED FINAL DIFFIE " + msg\_array.Length.ToString()+" " + msg\_array[2] + " " + msg\_array[3]);

Communicator.HandleResponse(msg\_array[2]);

Communicator.SetToClient();

this.NotepadButton.Enabled = true;

this.TiTaToButton.Enabled = true;

}

break;

case "/Mouse\_pos":

{

int m = 0, n = 0;

m = Int32.Parse(msg\_array[2]);

n = Int32.Parse(msg\_array[3]);

pictureBox.BringToFront();

pictureBox.Location = new Point(m, n - 181);//Set the other client's mouse position.

}

break;

case "/OpenNP":

{

try

{

Communicator.AddForm(Np);

this.BeginInvoke((Action)(Np.Show));

flag = true;

this.NotepadButton.Enabled=false;

}

catch

{

Console.WriteLine("Form failed. Restart the program.");

}

}

break;

case "/CloseNP":

{

this.NotepadButton.Enabled = true;

Console.WriteLine("Enabled: " + this.NotepadButton.Enabled.ToString());

Console.WriteLine("Flag: " + flag);

if (flag)

{

Np = new Notepad();

Communicator.SendMessage(this.Name, "/CloseNP ");

flag = false;

this.NotepadButton.Enabled = true;

}

}

break;

case "/OpenTicTacToe":

{

if (!flag)

{

tic = new TicTacToe(TicValue.O);

this.TiTaToButton.Enabled = false;

Communicator.AddForm(tic);

this.Invoke((Action)(tic.Show));

flag = true;

}

}

break;

case "/CloseTicTacToe":

{

if (flag)

{

Communicator.SendMessage(this.Name, "/CloseTicTacToe");

this.TiTaToButton.Enabled = true;

flag = false;

}

}

break;

case "/OpenPf":

{

try

{

if (!flag)

{

Pf.Dispose();

Pf = new Paint();

Communicator.AddForm(Pf);

this.BeginInvoke((Action)(Pf.Show));

flag = true;

this.PaintButton.Enabled = false;

}

}

catch

{

Console.WriteLine("Form failed. Restart the program.");

}

}break;

case "/ClosePf":

{

if (flag)

{

Pf.Close();

Communicator.RemoveForm(Pf);

Communicator.SendMessage(this.Name, "/ClosePf ");

flag = false;

this.PaintButton.Enabled = true;

}

}

break;

default:

{

MessageBox.Show("Unexpected exception.");

}

break;

}

}

catch (ArgumentNullException e)

{

Console.WriteLine(e.Message);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

}

}

/// <summary>

/// Opens the notepad form whenever the notepad button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void NotepadButton\_click(object sender, EventArgs e)

{

Communicator.AddForm(Np);

Communicator.SendMessage(this.Name,"/OpenNP ");

Np.Show();

flag = true;

this.NotepadButton.Enabled = false;

}

/// <summary>

/// Removes the form from communicator and notify the server whenever this form is being closed by the client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void mainForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Communicator.RemoveForm(this);

Communicator.SendMessage("Exit");

Communicator.DisConnect();

}

/// <summary>

/// Opens the tictactoe form whenever the notepad button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

tic = new TicTacToe(TicValue.X);

Communicator.AddForm(tic);

Communicator.SendMessage(this.Name, "/OpenTicTacToe ");

tic.Show();

flag = true;

this.TiTaToButton.Enabled = false;

}

/// <summary>

/// Notify the other client about resizing this window.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void mainForm\_Resize(object sender, EventArgs e)

{

if (!flag)

{

if (i % 4 == 0)

{

Communicator.SendMessage(this.Name, ("/Window\_size " + this.Height.ToString() + " " + this.Width.ToString()));

}

i += 1;

}

}

/// <summary>

/// Opens the paint form whenever the notepad button is clicked.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

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

{

Pf.Dispose();

Pf = new Paint();

Communicator.AddForm(Pf);

Communicator.SendMessage(this.Name, "/OpenPf");

Pf.Show();

flag = true;

this.PaintButton.Enabled = false;

}

}

}

**Notepad.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using System.Net.Sockets;

using System.Threading;

namespace WindowsFormsApplication1

{

public partial class Notepad : baseForm

{

int i = 0; // access controls

bool flag = false; // access controls

/// <summary>

/// Creates the form.

/// </summary>

public Notepad()

{

InitializeComponent();

}

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

{

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array != null)

{

try

{

flag = true;

switch (msg\_array[1]) // look how chunky it is, lets take these 'case's to some cool private methods! :)

{

case "/Window\_size":

{

flag = false;

this.Height = int.Parse(msg\_array[2]);

this.Width = int.Parse(msg\_array[3]);

}

break;

case "/Text\_Change":

{

richTextBox1.Text = msg\_array[2].Replace('%', ' ');

}

break;

case "/Close":

{

Close();

}

break;

default:

{

MessageBox.Show("Unexpected exception.");

}

break;

}

flag = false;

}

catch (NullReferenceException ex1)

{

Console.WriteLine(ex1.Message);

}

catch (ArgumentNullException e)

{

Console.WriteLine(e.Message);

}

catch (Exception ex)

{

Console.WriteLine("Unexpected exception : {0}", ex.ToString());

}

}

}

/// <summary>

/// Notify the other client about resizing this window.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Notepad\_Resize(object sender, EventArgs e)

{

if (!flag)

{

if (i % 4 == 0)

{

richTextBox1.Height = -(this.Top - this.Bottom); // Set the textbox's height according to the window's.

richTextBox1.Width = (this.Right - this.Left); // Set the textbox's width according to the window's.

Communicator.SendMessage(this.Name, ("/Window\_size " + richTextBox1.Height.ToString() + " " + richTextBox1.Width.ToString()));

}

i += 1;

}

}

/// <summary>

/// Update the other client whenever there is a change in the text.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void richTextBox1\_TextChanged( object sender, EventArgs e )

{

if (!flag)

if (richTextBox1.Text.Length < 4060) // Limit to 4060 because of the message maximum bytes length.

{

Communicator.SendMessage(this.Name, ("/Text\_Change " + richTextBox1.Text.Replace(' ', '%')));//!!!!! <- I would have changed that to 0 or something

}

else

{

MessageBox.Show("You can't write anymore. The limit is 4060 characters.");

richTextBox1.Text = string.Join("",richTextBox1.Text.Take(4059).ToArray());

}

}

/// <summary>

/// Removes the form from communicator and notify the server whenever this form is being closed by the client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void Notepad\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!flag)

{

Communicator.SendMessage(this.Name, "/Close ");

Communicator.SendMessage("mainForm", "/CloseNP ");

Communicator.RemoveForm(this);

}

}

}

}

**Paint.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

/// <summary>

/// A form where the clients can use Paint application.

/// </summary>

public partial class Paint : baseForm

{

Graphics g; //access controls

Point sp = new Point(0,0); // starting point

Pen pen = new Pen(Color.Black, 1); // pen of this client

Pen pen2 = new Pen(Color.Black, 1); // pen of the other client

Point ep = new Point(0, 0); // end point

int k = 0; //access controls

bool flag = false; //access controls

/// <summary>

/// Creates the form.

/// </summary>

public Paint()

{

InitializeComponent();

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

this.MinimizeBox = false;

this.MaximizeBox = false;

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array == null)

{

}

else

{

switch (msg\_array[1])

{

case "Paint":

{ // "Paint SPX,SPY EPX,EPY Color Width"

Console.WriteLine(string.Join("/", msg\_array));

for (int i = 0; i < msg\_array.Length; i++)

{

Console.WriteLine("msg\_array[" + i.ToString() + "] :" + msg\_array[i]);

}

sp = new Point(int.Parse(msg\_array[2].Split(',')[0]), int.Parse(msg\_array[2].Split(',')[1]));

ep = new Point(int.Parse(msg\_array[3].Split(',')[0]), int.Parse(msg\_array[3].Split(',')[1]));

pen2 = new Pen(Color.FromName(msg\_array[4]), float.Parse(msg\_array[5]));

g = this.CreateGraphics();

g.DrawLine(pen2, sp, ep);

BringAllToFront();

}

break;

case "/Clear":

{

g = this.CreateGraphics();

g.Clear(Color.White);

}

break;

case "/Close":

{

Close();

}

break;

}

}

}

/// <summary>

/// This event will run once the form is loaded.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void PaintForm\_load(object sender, EventArgs e)

{

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = Color.Black;

pen.Width = 1;

DefaultColorBox.BackColor = Color.Black;

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = RedColorBox.BackColor;

pen.Width = 1;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = BlueColorBox.BackColor;

pen.Width = 1;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = GreenColorBox.BackColor;

pen.Width = 1;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = YellowColorBox.BackColor;

pen.Width = 1;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = OrangeColorBox.BackColor;

pen.Width = 1;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Update variable k to 1 whenever mouse is down.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void PaintForm\_MouseDown(object sender, MouseEventArgs e)

{

sp = e.Location; // Set the first starting point.

if (e.Button == MouseButtons.Left)

k = 1; //Means - Mouse is down.

}

/// <summary>

/// Updates k to 0 whenever mouse is back up again.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void PaintForm\_MouseUp(object sender, MouseEventArgs e)

{

k = 0; // Means- Mouse is up.

}

/// <summary>

/// Whenever mouse is down and moves it draws on the board.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void PaintForm\_MouseMove(object sender, MouseEventArgs e)

{

if(k==1) // Mouse is down.

{

ep = e.Location; // Set the second point to connect.

g = this.CreateGraphics();

g.DrawLine(pen, sp, ep); //Draws the line.

BringAllToFront(); // Brings all color boxes to front.

Communicator.SendMessage(this.Name, "Paint " + sp.X.ToString() + "," + sp.Y.ToString() + " " + ep.X.ToString() + "," + ep.Y.ToString() + " " + pen.Color.ToKnownColor().ToString() +" "+ pen.Width.ToString());

Communicator.SendMessage(this.Name, "Paint " + sp.X.ToString() + "," + sp.Y.ToString() + " " + ep.X.ToString() + "," + ep.Y.ToString() + " " + pen.Color.ToKnownColor().ToString() + " " + pen.Width.ToString());

Communicator.SendMessage(this.Name, "Paint " + sp.X.ToString() + "," + sp.Y.ToString() + " " + ep.X.ToString() + "," + ep.Y.ToString() + " " + pen.Color.ToKnownColor().ToString() + " " + pen.Width.ToString());

}

sp = ep; // Set the first point to connect.

}

/// <summary>

/// Brings all color boxes to front.

/// </summary>

private void BringAllToFront()

{

panel1.BringToFront();

OrangeColorBox.BringToFront();

BlueColorBox.BringToFront();

RedColorBox.BringToFront();

BlackColorBox.BringToFront();

DefaultColorBox.BringToFront();

GreenColorBox.BringToFront();

YellowColorBox.BringToFront();

}

/// <summary>

/// Make the necessary changes whenever this button is clicked. (White color).

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = Color.White;

pen.Width = 10;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Clears the whole paint board.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

g.Clear(Color.White);

Communicator.SendMessage(this.Name, "/Clear");

}

/// <summary>

/// Notify the other client about resizing this window.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void PaintForm\_Resize(object sender, EventArgs e)

{

}

/// <summary>

/// Make the necessary changes whenever this color box is clicked.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

pen.Color = BlackColorBox.BackColor;

pen.Width = 1;

DefaultColorBox.BackColor = pen.Color;

}

/// <summary>

/// Removes the form from communicator and notify the server whenever this form is being closed by the client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void PaintForm\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!flag)

{

Communicator.SendMessage(this.Name, "/Close ");

Communicator.SendMessage("mainForm", "/ClosePf ");

Communicator.RemoveForm(this);

}

}

}

}

**TicTacToe.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class TicTacToe : baseForm

{

int i = 0; // access controls

bool flag = false; // access controls

TicValue id; // The client's X or O value

Button[,] board; // The game board.

Bitmap Xphoto, Ophoto; // access controls

/// <summary>

/// Creates the form.

/// </summary>

/// <param name="newid"> X or O</param>

public TicTacToe(TicValue newid)

{

InitializeComponent();

id = newid;

Xphoto = Properties.Resources.X;

Ophoto = Properties.Resources.O;

}

/// <summary>

/// Creates the form.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parametere.</param>

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

{

board = new Button[3, 3]; //Initializes the board.

Build();

if (id.Equals(TicValue.O))

{

Enablers(false);

}

SetSize(this.Height, this.Width);

SetLocation();

}

/// <summary>

/// click event of all of the 9 buttons.

/// </summary>

/// <param name="sender">event parameter.</param>

/// <param name="e">event parameter.</param>

private void button\_click(object sender , EventArgs e)

{

Button b = sender as Button; // The button clicked.

b.Text= id.ToString();

int row = b.TabIndex / 3;

int column = b.TabIndex % 3;

if (id == TicValue.O)

b.Image = Ophoto;

if (id == TicValue.X)

b.Image = Xphoto;

Communicator.SendMessage(this.Name, ("/Turn "+row.ToString() +" "+ column.ToString()));

Turn(id, row, column);

Enablers(false);

if (IsFinished())

{

string winSide = WhoWon();

if (winSide.Equals(TicValue.E.ToString()))

winSide = "this is a tie.";

MessageBox.Show("Game is over because " + winSide+ " won the game.\r\n TicTacToe window will be closed now.");

Communicator.SendMessage("mainForm", "/CloseTicTacToe ");

Communicator.RemoveForm(this);

Close();

}

}

/// <summary>

/// Sets the enabled property of each button to the boolean given variable.

/// </summary>

/// <param name="x"> The enabled property value.</param>

private void Enablers(bool x)

{

for (int j = 0; j < board.GetLength(0); j++)

{

for (int k = 0; k < board.GetLength(1); k++)

{

board[j, k].Enabled=x;

}

}

}

/// <summary>

/// Initializes each and every board tile.

/// </summary>

private void Build()

{

for (int j = 0; j < board.GetLength(0); j++)

{

for (int k = 0; k < board.GetLength(1); k++)

{

board[j, k] = new Button();

this.Controls.Add(board[j, k]); //Add the buttons to the system's controls.

board[j,k].UseVisualStyleBackColor = true;

board[j,k].Click += new EventHandler(button\_click);

board[j, k].Text = TicValue.E.ToString();

board[j, k].TabIndex = 3 \* j + k;

board[j, k].Show();

}

}

}

/// <summary>

/// Set the size of all the buttons.

/// </summary>

/// <param name="Height">Height value.</param>

/// <param name="Width">Width value.</param>

private void SetSize(int Height, int Width)

{

for (int j = 0; j < board.GetLength(0); j++)

{

for (int k = 0; k < board.GetLength(1); k++)

{

board[j, k].Height = (Height) / 3;

board[j, k].Width = Width / 3;

}

}

}

/// <summary>

/// Sets the buttons' location according to the window's size.

/// </summary>

private void SetLocation()

{

for (int j = 0; j < board.GetLength(0); j++)

{

for (int k = 0; k < board.GetLength(1); k++)

{

board[j, k].Location = new Point(j \* (this.Width / 3), k \* ((this.Height-51) / 3));

}

}

}

/// <summary>

/// Notify the other client about resizing this window.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TicTacToe\_Resize(object sender, EventArgs e)

{

if (!flag)

{

if (i % 4 == 0) //Efficiency.

{

SetSize(this.Height-51, this.Width);

Communicator.SendMessage(this.Name, ("/Window\_size " + this.Height.ToString() + " " + this.Width.ToString()));

}

i += 1;

}

SetLocation();

}

/// <summary>

/// Updates the board according to the turn.

/// </summary>

/// <param name="currID"> X's/O's turn.</param>

/// <param name="row"> row pressed.</param>

/// <param name="column"> column pressed.</param>

public void Turn(TicValue currID, int row, int column)

{

board[row, column].Text = currID.ToString();

if (currID == TicValue.O)

board[row, column].Image = Ophoto;

if(currID==TicValue.X)

board[row, column].Image = Xphoto;

}

/// <summary>

/// Calculates whether the specified row is all X or all O

/// </summary>

/// <param name="rowNum">row number.</param>

/// <returns></returns>

internal bool IsWinRow(int rowNum)

{

if (this.board[rowNum, 0].Text.Equals(this.board[rowNum, 1].Text) && this.board[rowNum, 0].Text.Equals(this.board[rowNum, 2].Text) && (! this.board[rowNum, 0].Text.Equals(TicValue.E.ToString())))//first row

return true;

return false;

}

/// <summary>

/// Calculates whether the specified column is all X or all O

/// </summary>

/// <param name="columnNum">column number.</param>

/// <returns></returns>

internal bool IsWinColumn(int columnNum)

{

if (this.board[0, columnNum].Text.Equals(this.board[1, columnNum].Text) && this.board[0, columnNum].Text.Equals(this.board[2, columnNum].Text) && !this.board[0, columnNum].Text.Equals(TicValue.E.ToString()))//first column

return true;

return false;

}

/// <summary>

/// Calculates whether the specified diagonal is all X or all O

/// </summary>

/// <param name="diagonalNum">diagonal number (0- main, 1-secondary).</param>

/// <returns></returns>

internal bool IsWinDiagonal(int diagonalNum)

{

if(diagonalNum==0)

if (this.board[0, 0].Text.Equals(this.board[1, 1].Text) && this.board[0, 0].Text.Equals(this.board[2, 2].Text) && !this.board[0, 0].Text.Equals(TicValue.E.ToString()))//main diagonal line

return true;

if (diagonalNum==1)

if (this.board[0, 2].Text.Equals(this.board[1, 1].Text) && this.board[0, 2].Text.Equals(this.board[2, 0].Text) && !this.board[1, 1].Text.Equals(TicValue.E.ToString()))//second diagondal line

return true;

return false;

}

/// <summary>

/// Calculates whether someone won the game already.

/// </summary>

/// <returns> boolean true if finished, false otherwise.</returns>

public bool IsFinished()

{

bool x = false;

for (int i = 0; i < 3; i++)

{

x = x || IsWinRow(i);

x = x || IsWinColumn(i);

x = x || IsWinDiagonal(i);

}

if (x)

return x;

///No one has won so far.

///Checks for a tie.

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

if (this.board[i, j].Text.Equals(TicValue.E.ToString()))

return false;

}

}

return true; // All tiles are not empty

}

/// <summary>

/// Returns the winning sign. (E if it's a tie).

/// </summary>

/// <returns> string contains the winning sign.</returns>

public string WhoWon()

{

if (this.board[0, 0].Text.Equals(this.board[0, 1].Text) && this.board[0, 0].Text.Equals(this.board[0, 2].Text))//first row

return this.board[0, 0].Text;

if (this.board[1, 0].Text.Equals(this.board[1, 1].Text) && this.board[1, 0].Text.Equals(this.board[1, 2].Text))//second row

return this.board[1, 0].Text;

if (this.board[2, 0].Text.Equals(this.board[2, 1].Text) && this.board[2, 0].Text.Equals(this.board[2, 2].Text))//third row

return this.board[2, 0].Text;

if (this.board[0, 0].Text.Equals(this.board[1, 0].Text) && this.board[0, 0].Text.Equals(this.board[2, 0].Text))//first column

return this.board[0, 0].Text;

if (this.board[0, 1].Text.Equals(this.board[1, 1].Text) && this.board[0, 1].Text.Equals(this.board[2, 1].Text))//second column

return this.board[0, 1].Text;

if (this.board[0, 2].Text.Equals(this.board[1, 2].Text) && this.board[0, 2].Text.Equals(this.board[2, 2].Text))//third column

return this.board[0, 2].Text;

if (this.board[0, 0].Text.Equals(this.board[1, 1].Text) && this.board[0, 0].Text.Equals(this.board[2, 2].Text))//main diagonal line

return this.board[0, 0].Text;

if (this.board[0, 2].Text.Equals( this.board[1, 1].Text) && this.board[0, 2].Text.Equals( this.board[2, 0].Text) )//secondary diagondal line

return this.board[0, 2].Text;

return TicValue.E.ToString();

}

/// <summary>

/// Removes the form from communicator and notify the server whenever this form is being closed by the client.

/// </summary>

/// <param name="sender"> event parameter.</param>

/// <param name="e"> event parameter.</param>

private void TicTacToe\_FormClosing(object sender, FormClosingEventArgs e)

{

if (!flag)

{

Communicator.SendMessage(this.Name, "/Close ");

Communicator.SendMessage("mainForm", "/CloseTicTacToe ");

Communicator.RemoveForm(this);

}

}

/// <summary>

/// Recieve method that process the input.

/// </summary>

/// <param name="msg\_array"> The input seperated by ' '.</param>

public override void Recieve(string[] msg\_array)

{

if (msg\_array == null)

{

}

else

{

try

{

switch (msg\_array[1])

{

case "/Window\_size":

{

flag = false;

this.Height = int.Parse(msg\_array[2]);

this.Width = int.Parse(msg\_array[3]);

}

break;

case "/Turn":

{

if (id == TicValue.O)

{

Turn(TicValue.X, int.Parse(msg\_array[2]), int.Parse(msg\_array[3]));

}

else

Turn(TicValue.O, int.Parse(msg\_array[2]), int.Parse(msg\_array[3]));

if (IsFinished())

{

string winSide = WhoWon();

if (winSide.Equals(TicValue.E.ToString()))

winSide = "this is a tie.";

MessageBox.Show("Game is over because " + winSide + " won the game.\r\n TicTacToe window will be closed now.");

Communicator.SendMessage("mainForm", "/CloseTicTacToe ");

Communicator.RemoveForm(this);

Close();

}

for (int j = 0; j < board.GetLength(0); j++)

{

for (int k = 0; k < board.GetLength(1); k++)

{

if (board[j, k].Text.Equals(TicValue.E.ToString()))

board[j, k].Enabled = true;

else

continue;

}

}

}

break;

case "/Close":

{

Close();

}break;

}

}

catch (FormatException e)

{

Console.WriteLine(e.Message);

}

catch (ArgumentNullException e)

{

Console.WriteLine(e.Message);

}

catch (NullReferenceException e)

{

Console.WriteLine(e.Message);

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}

}