

מגיש: יהונתן טשבינר

כיתה: יב'6

בית ספר: תיכון אלון רמת השרון

שם המורה: עידן פלדברג

**פרוייקט eMessage**

**מסמך אפיון – יהונתן טשבינר יב'6**

**גרסה סופית 5**

**25.6.2018**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 8.12 | 1 | מידע ראשוני אודות הפרוייקט |
| 2.1 | 2 | הוספת מידע |
| 5.1 | 3 | רישום המידע נוסף |
| 5.6 | 4 | רישום מידע נוסף |
| 25.6 | 5 | רישום מידע סופי בעקבות שינויים משמעותיים במוצר |

**1.הקדמה**

פרויקט זה עומד לדמות בנראות ובצורת הפעולה את WhatsApp. זהו מוצר שבו מתבצע צ'אט של שני מחשבים (תומך גם ביותר משתמשים אך נועד לשימוש של שני משתמשים בלבד), שבו ניתן לשלוח הודעות טקסט, אימוג'ים, ותמונות התומך בשפות העברית והאנגלית.

**1.1 מטרה**

מטרת הפרויקט היא ליצור אמצעי תקשורת חדש בין אנשים באמצעות המחשבים, וקהל היעד שלו הוא כלל הגילאים.

**1.2 המוצר**

* שם המוצר: eMessage
* הסבר כללי: במוצר מתבצע צ'אט בין שני מחשבים, שבו ניתן לשלוח הודעות טקסט ותמונות ממשתמש למשתמש.
* מטרות המוצר ויתרונותיו: מטרת המוצר היא ליצור אמצעי תקשורת חדש ויעיל בין אנשים באמצעות המחשבים. המוצר הפונה לכלל הגילאים והיתרון בו הוא שאין צורך להזין את מספר הטלפון שלכם, פשוט להתחבר עם שם משתמש וסיסמה ולהיכנס לחדר הצ'אט, שם תצטרכו להמתין או שימתין לכם כבר שם משתמש נוסף.

**1.3 הגדרות**

Tkinter – ממשק משתמש גרפי סטנדרטי (GUI)

Socket – מודול לממשק עם תכנות רשת

Threading – מאפשר את ההדדיות בצ'אט בין המשתמשים, כלומר: באמצעותו משתמש אחד יכול לשלוח הודעה וישר לאחר מכן המשתמש השני יקבל אותה.

PIL – מאפשר העלאת תמונות ושימוש בהן לפונקציות שונות בהתאם לרצונות האישיים של יוצר הפרוייקט.

Jpg - פורמט דחיסה גרפי בקובצי מחשב לדחיסת תמונות

Python – שפת תכנות

Base64 – מודול האחראי על שליחת התמונות ממשתמש אחד לשני, בכך שהוא ממיר את המידע אשר נקרא מן התמונה למחרוזת מסוג מודול זה ופועלת בהתאם למען יצירת התמונה לשני המשתמשים.

os – מודול האחראי על טיפול בשגיאות העלולות להתרחש בקוד, באמצעות יציאה באופן מסודר של המשתמש מהתוכנית.

**1.4 תקציר**

בהמשך מסמך זה יופיעו הסברים כלליים אודות הפרויקט, מה הוא יכיל לפי הקריטריונים הדרושים להגשת המסמך ובסדר הנכון.

**2. תיאור כללי**

**2.1 פונקציונליות**

הפרוייקט מופעל דרך התוכנה Python דרך הממשק Tkinter, תוכנת עיצוב GUI (ממשק משתמש גרפי) שפועלת דרך התוכנה Python, שדרכה הפרוייקט נכתב. בנוסף, התקשורת בין המחשבים מתבצעת דרך Socket שהוא מודול לממשק עם תכנות רשת.

**2.2 קהל יעד**

קהל היעד לפרוייקט אינו ספציפי. כלומר, הקהל יכול להיות ילדים, נערים, מבוגרים, קשישים, ובקיצור, כולם יכולים לבצע שימוש בפרוייקט. השימוש בתוכנה הוא בסיסי ביותר ואין צורך בידע מוקדם על מנת לבצע שימוש בה, פשוט להזין את שם המשתמש הרצוי ואת הסיסמה (אשר ידועה מראש לשני המשתמשים טרם כניסתם לתוכנה), וללחוץ "Login", ועל מנת להקליד הודעות נדרש אך ורק ללחוץ על כפתור ה-"Send" או מקש ה-Enter, או במידה והמשתמש רוצה להעלות תמונה, עליו ללחוץ על כפתור ה"Upload", לאחר מכן יתבקש לבחור תמונה וללחוץ על כפתור האישור. בנוסף אם ברצונו לשלוח הודעת Emoji, עליו בסך הכל ללחוץ על הכפתור אשר מציג תמונה של הEmoji הרצוי לפיו.

**2.3 אילוצים עיקריים**

מערכת הפרוייקט דורשת חיבור בין שני מחשבים. במידה ושני המחשבים נמצאים על אותה הרשת, אין צורך בחיבור לאינטרנט. במידה ולא, יש צורך של שני המחשבים בחיבור לאינטרנט

בנוסף, דרוש Python בגרסה 2.7 ומעלה.

**2.4 הנחות ותלויות**

המערכת תעבוד בסביבת windows. עם זאת ייתכן והמערכת לא תעבוד בסביבות ריצה (מערכות הפעלה) אחרות, כגון : linux, mac.

**2.5 מצב השוק כיום**

כיום המוצר בעל השימוש הניכר ביותר לצורכי צ'אט במחשב הוא WhatsApp.net, אשר נראה כמעט זהה לאפליקציה בטלפון החכם, שבו ניתן לבצע את אותן הפעולות כפי שניתן לבצע דרך האפליקציה בטלפון. ההתחברות לצ'אט נעשית באמצעות מספרי טלפון, בזמן שההתחברות לצ'אט אשר הוכן על-ידי, נעשית באמצעות סיסמה.

**3. דרישות מפורטות**

**3.1 דרישות פונקציונליות**

התחברות: בודק האם שם המשתמש קיים במערכת והאם הסיסמה נכונה.

|  |  |  |
| --- | --- | --- |
|  | **שרת** | **לקוח** |
| **קלט** | אין | שם משתמש וסיסמה |
| **פלט** | הודעה מסוג משתנה בוליאני המאשר/לא מאשר התחברות בהתאם לנתונים שהזין המשתמש | * במידה וההתחברות התבצעה בהצלחה מעבר לחדר הצ'אט * במידה וההתחברות אינה התבצעה בהצלחה, תוצג השגיאה על פני מסך ההתחברות ויוכל לנסות להתחבר בשנית. |
| **טיפול בשגיאות** | * בדיקת תקינות המידע שהגיע מהלקוח (תווים מורשים, אורך שם המשתמש, סיסמה נכונה). * בדיקה האם שם המשתמש קיים * סגירת אחד מן הלקוחות באופן רצוי או בלתי צפוי | * טיפול בשגיאת חיבור לשרת * סגירת השרת באופן בלתי צפוי |

שליחה הודעה: שליחת הודעה לכלל המשתמשים המחוברים לשרת.

|  |  |  |
| --- | --- | --- |
|  | **שרת** | **לקוח** |
| **קלט** | אין | * מקלדת: הודעת טקסט * כפתורי האימוג'ים: הודעת אימוג'י. * כפתור העלאת התמונה: הודעת תמונה |
| **פלט** | אין | * ההודעה מוצגת על מסך חדר הצ'אט בהתאם לסוגה ולאיזה מן הלקוחות שלח את ההודעה |
| **טיפול בשגיאות** | * בדיקת תקינות המידע שהגיע מהלקוח (תווים מורשים, אורך שם המשתמש, סיסמה נכונה). * בדיקה האם שם המשתמש קיים * סגירת אחד מן הלקוחות באופן רצוי או בלתי צפוי | * טיפול בשגיאת חיבור לשרת * סגירת השרת באופן בלתי צפוי |

**3.2 דרישות של ממשקים חיצוניים**

**3.2.1 ממשקי משתמש**

אני משתמש בתוכנת הTkinter שיכולה להפעיל GUI (ממשק משתמש גרפי סטנדרטי) , אשר באמצעותה ניתן לייצר את הגרפיקה והעיצוב של הפרוייקט עצמו.

**3.2.2 ממשקי חומרה**

מחשב, עכבר, מקלדת, מסך.

**3.2.3 ממשקי תוכנה**

בפרויקט זה יש צורך בתקשור בין תוכנות שונות. יש תוכנה המותאמת ללקוח ותוכנה המותאמת לשרת שיכולות השימוש של כל אחד מהם שונות.

**3.2.4 ממשקי תקשורת**

בפרוייקט זה חלה תקשורת בין מחשבים באמצעות Socket (אמצעי תקשורת בין רשתות), שבו מחשב אחד משמש בתור לקוח והשני משמש בתור שרת.

על מנת לבצע תקשורת זו על המחשבים: להיות באותה הרשת, על מנת לוותר על שימוש באינטרנט. ובמידה ולא, על שני המחשבים להיות בעלי חיבור לאינטרנט.

**3.3 דרישות לא פונקציונליות**

**3.3.1 דרישות ביצועים (performance)**

הקוד מורץ יהיה ניתן להזין כל שם משתמש אפשרי, ותינתן האפשרות לשלוח כל הודעה האפשרית מסוג: טקסט או תמונה/אימוג'י, ללא קריסות מיותרות או חריגות.

אין דרישות לגבי מפרט המחשב על מנת הרצת הפרוייקט, אך ורק גרסת Python 2.7 ומעלה.

**3.3.2 דרישות מהימנות (reliability)**

שליחת קבצי תמונות מפורמט jpg, והעברתן בצורה חלקה ומלאה למשתמש השני אשר גם הוא נמצא בחדר הצ'אט.

בנוסף,

**3.3.3 דרישות זמינות (availability)**

על הפרוייקט לבצע את ההפעלה שלו תוך זמן קצר ביותר בין פעולה לפעולה. למשל: כאשר המשתמש מזין את שם המשתמש הרצוי לו, וה-IP ולוחץ על הכפתור "התחבר", כאשר המשתמש השני מבקש להתחבר בזמן שהזין את אותו ה-IP, על חלון אימות הנתונים להיסגר מידית ושחלון הצ'אט יפתח בזמן מהיר מאוד, ושבמהלך ביצוע פעולות אלו במידה ושני המחשבים אינם נמצאים על אותה הרשת, שיהיה חיבור טוב לאינטרנט שגם יוכל להשפיע על מהירות ביצוע הפעולות.

**3.3.4 דרישות אבטחה (security)**

הכנסת שם משתמש, על מנת לזהות את שם כותב ההודעות בחדר הצ'אט.

הכנסת סיסמה, על מנת ליצור את ההתחברות עם שרת ספציפי.

**3.3.5 דרישות תחזוקה (maintainability)**

במהלך שליחת הודעות הטקסט והתמונות/אימוג'ים, כל הדברים אשר ניתנים לשליחה ישלחו במלואם אל המחשב השני ולא ייאבדו בדרך ולא יתכן מצב כי חלק מתכולתם תימחק או תיפגע. כל הדברים האלו יכולים להגיע למחשב השני במלואם לאחר שליחת ההודעה.

**3.3.6 דרישות ניידות (portability)**

האפשרות לבצע את פעולת הפרויקט במלואה אך ורק על מחשב נייד או נייח.

**3.4 דרישות בסיס נתונים**

הנתונים אשר יישמרו הם התמונות, אשר נשלחים ממחשב אחד למחשב אחר יוצגו על מסך חדר הצ'אט בהתאם בלבד.

למשל: כאשר משתמש 1 ירצה לשלוח תמונה למשתמש 2, תוצג התמונה על פני המסך של משתמש 1 והיא תישלח בו בזמן למשתמש 2 והיא תוצג על המסך שלו גם בהתאם, אך לא תישמר על המחשב שלו ולהיפך.

**<eMessage>**

מסמך עיצוב

<יהונתן טשבינר>

<גרסה 4>

<26.06.2018>

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 6.6 | 1 | הוספת המידע אשר יש אודות הפרויקט עד כה |
| 7.6 | 2 | הוספת פרטים נוספים |
| 25.6 | 3 | הוספת שינויים ניכרים אודות המוצר |
| 26.6 | 4 | הוספת שינויים מינוריים ואחרונים |



**הקדמה**

מטרה

מסמך זה הוא מסמך העיצוב של פרויקט הגמר בסייבר בהיקף של 5 יחידות לימוד. מטרת מסמך זה הוא לספק מידע אודות ארכיטקטורת המערכת, ממה היא מורכבת, המודולים בה היא משתמשת, צורת הפעולה שלה והעיצוב שלה בכדי שהקורא יבין באופן נרחב אודות המוצר. קהל היעד העיקרי של המסמך הינם המורה, עוזר ההוראה ובוחני הבגרות החיצוניים בסייבר.

המוצר

שם המוצר המפותח: eMessage

המוצר אשר בניתי הינו צ'אט בין שני משתמשים (יכול לעבוד גם כרב משתמשים אך אינו מתופעל לחלוטין לפעולה זו מבחינה גרפית) הפועל באמצעות תקשורת לקוח-שרת (client-server), בשימוש במודול socket. בנוסף שני המשתמשים יכולים לשלוח האחד לשני תמונות ואימוג'ים, והמוצר תומך בשפות אנגלית ועברית.

יתרונות: מאובטח באמצעות סיסמה המגובבת באמצעות md5 – הנועדה למניעת פריצות אל השרת ובעיות, נוח וקל לשימוש, נראות המוצר ידידותית למשתמש.

הגדרות

Tkinter – מודול האחראי על העיצוב והגרפיקה של המוצר. מודול זה הוא ממשק משתמש גרפי סטנדרטי (GUI), הנחשבת כנפוצה ביותר שבאמצעותה ניתן לעצב מגוון של מוצרים ב-Python.

Socket – מודול לממשק עם תכנות רשת, המתבסס על תקשורת של לקוח-שרת (client-server). במקרה זה, גם הלקוח וגם השרת הינם משתמשים, והמודול נכנס לפעולה כאשר יש ביקוש לחיבור בין שני המשתמשים,

Threading – מודול האחראי על ההדדיות בין ההודעות הנשלחות ממשתמש אחד אל השני בחדר צ'אט, כלומר: באמצעותו משתמש אחד יכול לשלוח הודעה וישר לאחר מכן המשתמש השני יקבל אותה. משתנה מסוג מודול זה מתמקד על פונקציה מסוימת ומריץ אותה באופן אינסופי בהתאם לרצונות המשתמש באופן שאינו מגביל את השימוש של התהליכים האחרים בקוד המוצר.

PIL – מודול האחראי על העלאת התמונות והצגתן במסך חדר הצ'אט. ניתן להשתמש בתמונות בפונקציות השונות בהתאם לרצונותיו האישיים של יוצר הפרויקט. כאשר משתמש אחד מעוניין לשלוח למשתמש השני התמונה, ראשית כל עליו לבחור איזו תמונה עליו להעלות ועל מנת שהיא תוצג על המסך ותוכל להישלח, מטרה זו מתממשת בזכות מודול זה.

tkMessageBox – מודול האחראי על הודעות קופצות (Pop-ups) אשר מסמנות שגיאות של המשתמש בעת התחברותו למערכת, למשל: שם משתמש שאינו תקין, או IP שאינו תקין. והודעות שאלה השואלות את המשתמש בעת לחיצתו על כפתור ה"Quit" או ה-X, האם הוא בטוח שהוא רוצה לצאת, ובאפשרותו לסמן כן או לא, במידה וסימן כן, פעילות התוכנית תסתיים לחלוטין ובמידה וסימן לא היא תמשיך מאותה הנקודה אשר היה בה.

tkFileDialog – מודול האחראי על החלון הנפתח למטרת העלאת תמונות אשר משתמש אחד מעוניין לשלוח למשתמש השני. באמצעות מודול זה למשתמש יש את האפשרות באמצעות חלון אשר נפתח באמצעות לחיצה על כפתור ה"Upload" בחדר הצ'אט, לבחור את התמונה שאותה הוא מעוניין לשלוח אל המשתמש שאליו הוא מחובר.

logging – מודול המשמש כתוסף אשר באמצעותו המידע אשר מודפס בקוד עצמו (לא בתוך מסך המוצר) מאוחסן בתוכו ומתחדש בכל פעם שהמוצר מופעל מחדש.

Base64 – מודול האחראי על שליחת התמונות ממשתמש אחד לשני, בכך שהוא ממיר את המידע אשר נקרא מן התמונה למחרוזת מסוג מודול זה ופועלת בהתאם למען יצירת התמונה לשני המשתמשים.

os – מודול האחראי על טיפול בשגיאות העלולות להתרחש בקוד, באמצעות יציאה באופן מסודר של המשתמש מהתוכנית.

**ארכיטקטורת המערכת**

**מבט על**

* המוצר נבנה במודל שרת-לקוח (client-server model)





**server**

**client**

****

**client**

**מודל שרת-לקוח :(client-server model)**

ארכיטקטורת תוכנה לחישוב מבוזר, אשר מגדירה את היחס בין תוכנות משתפות פעולה. המודל מחלק את המשימות או עומס העבודה בין ספק השירות או המשאבים - השרת, לבין מבקש השירות - הלקוח. שרת-לקוח היא אחת מתצורות ההתקשרות הנפוצות ברשתות מחשבים. השרת הוא תוכנה פסיבית, המאזינה לרשת ומחכה לקבל בקשות. הלקוח לעומתו בדרך כלל מהווה את ממשק המשתמש - הוא מופעל על ידי המשתמש ופונה לשרת כאשר הוא זקוק למידע או שירותים ממנו.

**לקוח (Client)**

הלקוח במוצר מייצג את אחד מן המשתמשים המעוניינים להתחבר לצ'אט. כאשר המשתמש מפעיל את המוצר דרך קוד הלקוח נפתח לו מסך הפתיחה, אשר מקנה בעבורו מספר אפשרויות וביניהן דרכי התחברות.

על מנת שהמשתמש יוכל להתחבר אל הצ'אט, עליו להזין שם משתמש כרצונו ואת הסיסמה אשר ידועה מראש לשני המשתמשים, ולאחר מכן על מנת לעבור לחדר הצ'אט על המשתמש ללחוץ על כפתור ה"Login". במידה ושם המשתמש והסיסמה שניהם תקינים, יכנס המשתמש לחדר הצ'אט ושם יוכל לשלוח הודעות כרצונו, איך אינן ישלחו למשתמש השני עד שהוא יתחבר.

מסך חדר הצ'אט נפתח כאשר יש חיבור בין הלקוח – המייצג משתמש לבין השרת.



**המודולים אשר בהם משתמש הלקוח:**

Tkinter – מודול האחראי על העיצוב והגרפיקה של המוצר. מודול זה הוא ממשק משתמש גרפי סטנדרטי (GUI), הנחשבת כנפוצה ביותר שבאמצעותה ניתן לעצב מגוון של מוצרים ב-Python. מודול זה אחראי על הנראות של התהליכים המתרחשים בתוך הקוד.

מודול זה עובד לעתים בשיתוף עם המודול PIL, בתהליך הצגת התמונה אשר הועלתה/התבקשה להיות מועלה על-ידי המשתמש/התקבלה על-ידי המשתמש השני.

Socket – מודול לממשק עם תכנות רשת, המתבסס על תקשורת של לקוח-שרת (client-server). במקרה זה, הלקוח מייצג אחד מן המשתמשים, והמודול נכנס לפעולה כאשר יש ביקוש מהלקוח לחיבור לשרת.

Threading – מודול האחראי על ההדדיות בין ההודעות הנשלחות ממשתמש אחד אל השני בחדר צ'אט, כלומר: באמצעותו הלקוח יכול לשלוח הודעה וישר לאחר מכן השרת יקבל אותה. משתנה מסוג מודול זה מתמקד על פונקציה מסוימת ומריץ אותה באופן אינסופי בהתאם לרצונות המשתמש באופן שאינו מגביל את השימוש של התהליכים האחרים בקוד המוצר.

בנוסף, מודול זה עובד בשיתוף עם Socket בתהליך קבלת ההודעות מן הלקוח. ישנה פונקציה בתוך המחלקה Client, אשר נוצר בתוכה Thread, המתמקד על פונקציה המקבלת מידע מן השרת.

PIL – מודול האחראי על העלאת התמונות והצגתן במסך חדר הצ'אט. ניתן להשתמש בתמונות בפונקציות השונות בהתאם לרצונותיו האישיים של יוצר הפרויקט. כאשר משתמש אחד מעוניין לשלוח למשתמש השני התמונה, ראשית כל עליו לבחור איזו תמונה עליו להעלות ועל מנת שהיא תוצג על המסך ותוכל להישלח, מטרה זו מתממשת בזכות מודול זה.

tkMessageBox – מודול האחראי על הודעות קופצות (Pop-ups) אשר מסמנות שגיאות של הלקוח בעת התחברותו למערכת, למשל: שם משתמש שאינו תקין, או IP שאינו תקין. והודעות שאלה השואלות את המשתמש בעת לחיצתו על כפתור ה"Quit" או ה-X, האם הוא בטוח שהוא רוצה לצאת, ובאפשרותו לסמן כן או לא, במידה וסימן כן, פעילות התוכנית תסתיים לחלוטין ובמידה וסימן לא היא תמשיך מאותה הנקודה אשר היה בה.

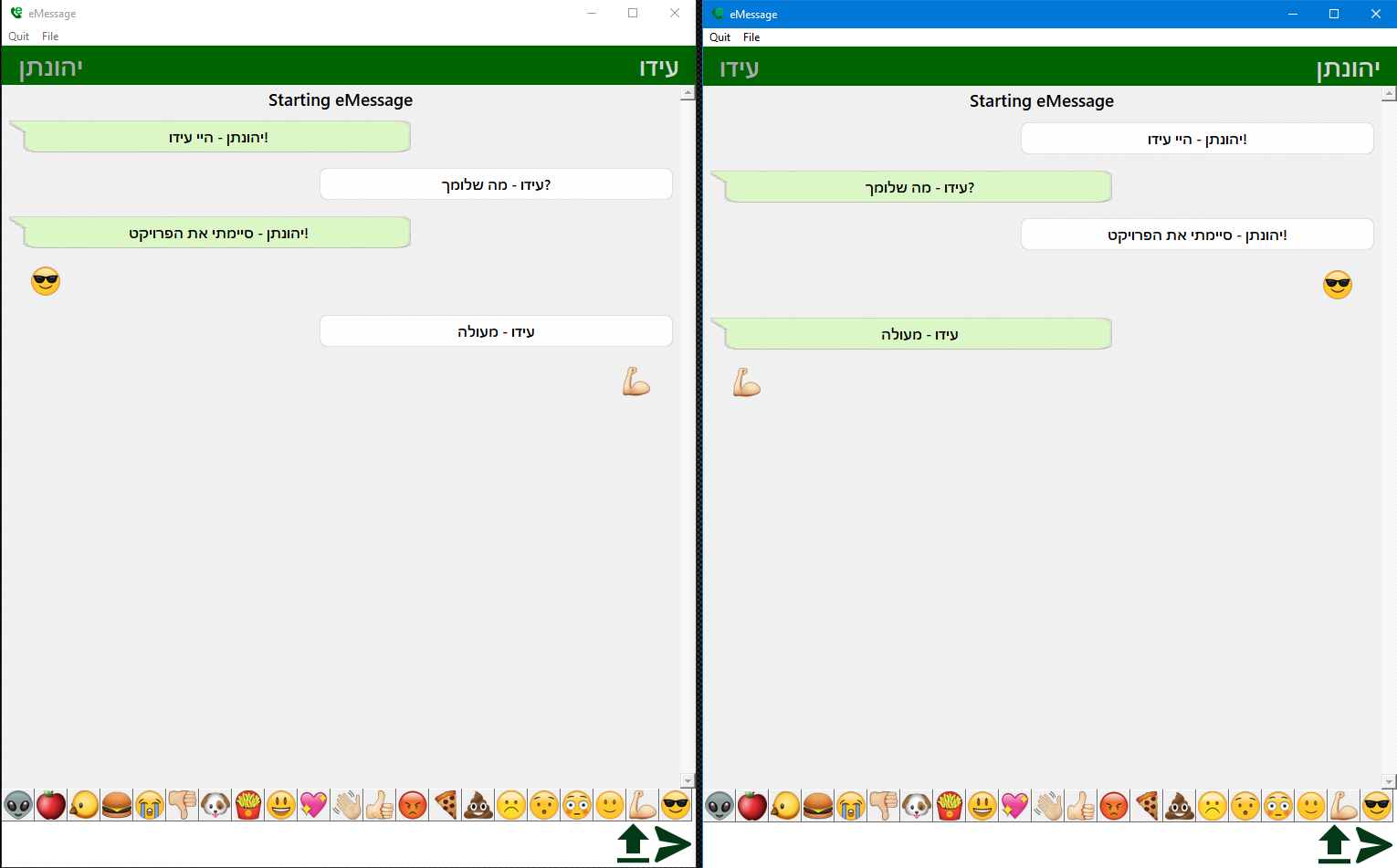
tkFileDialog – מודול האחראי על החלון הנפתח למטרת העלאת תמונות אשר משתמש אחד מעוניין לשלוח למשתמש השני. באמצעות מודול זה ללקוח יש את האפשרות באמצעות חלון אשר נפתח באמצעות לחיצה על כפתור ה"Upload" בחדר הצ'אט, לבחור את התמונה שאותה הוא מעוניין לשלוח אל השרת שאליו הוא מחובר.

logging – מודול המשמש כתוסף אשר באמצעותו המידע אשר מודפס בקוד עצמו (לא בתוך מסך המוצר) מאוחסן בתוכו ומתחדש בכל פעם שהמוצר מופעל מחדש.

Base64 – מודול האחראי על שליחת התמונות ממשתמש אחד לשני, בכך שהוא ממיר את המידע אשר נקרא מן התמונה למחרוזת מסוג מודול זה ופועלת בהתאם למען יצירת התמונה לשני המשתמשים.

os – מודול האחראי על טיפול בשגיאות העלולות להתרחש בקוד, באמצעות יציאה באופן מסודר של המשתמש מהתוכנית.

StringIO – מודול המספק אמצעי נוח לעבודה עם טקסט בזיכרון באמצעות ממשק ה- API של הקובץ (קריאה, כתיבה וכו').



**שרת (Server)**

השרת הינו אחראי על קיום פעילות התכנית של שני המשתמשים, אשר אמורים לבצע התחברות על השרת.

פעילות השרת מתחלקת למספר חלקים:

* קבלת הודעת טקסט/תמונה ממשתמש אחד, ושליחתן לשני המשתמשים (הלקוח יודע לטפל במקרה זה בהתאם).
* זיהוי פרטי ההתחברות של הלקוח וביצוע פעולות בהתאם (בדיקת הסיסמה – האם נכונה, בדיקת שם המשתמש – האם הינו נמצא בשימוש כעת).

**המודולים אשר בהם משתמש הלקוח:**

* Socket – מודול לממשק עם תכנות רשת, המתבסס על תקשורת של לקוח-שרת (client-server). במקרה זה, הלקוח מייצג אחד מן המשתמשים, והמודול נכנס לפעולה כאשר יש ביקוש מהלקוח לחיבור לשרת.
* Threading – מודול האחראי על ההדדיות בין ההודעות הנשלחות ממשתמש אחד אל השני בחדר צ'אט, כלומר: באמצעותו הלקוח יכול לשלוח הודעה וישר לאחר מכן השרת יקבל אותה. משתנה מסוג מודול זה מתמקד על פונקציה מסוימת ומריץ אותה באופן אינסופי בהתאם לרצונות המשתמש באופן שאינו מגביל את השימוש של התהליכים האחרים בקוד המוצר.

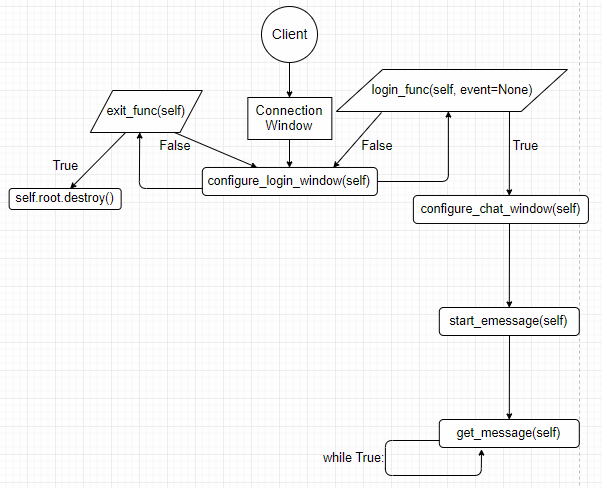
בנוסף, מודול זה עובד בשיתוף עם Socket בתהליך קבלת ההודעות מן הלקוח. ישנה פונקציה בתוך המחלקה Client, אשר נוצר בתוכה Thread, המתמקד על פונקציה המקבלת מידע מן השרת.

* logging – מודול המשמש כתוסף אשר באמצעותו המידע אשר מודפס בקוד עצמו (לא בתוך מסך המוצר) מאוחסן בתוכו ומתחדש בכל פעם שהמוצר מופעל מחדש.
* Base64 – מודול האחראי על שליחת התמונות ממשתמש אחד לשני, בכך שהוא ממיר את המידע אשר נקרא מן התמונה למחרוזת מסוג מודול זה ופועלת בהתאם למען יצירת התמונה לשני המשתמשים.
* os – מודול האחראי על טיפול בשגיאות העלולות להתרחש בקוד, באמצעות יציאה באופן מסודר של המשתמש מהתוכנית.
* hashlib – מודול האחראי על האבטחה של הסיסמה המוגדרת מראש בתוך השרת, באמצעות המרת הסיסמה הרצויה ל-md5, אשר ניתן להמרה אליו אך לא ממנו. מטרת ההצפנה היא מניעת פריצות על-ידי משתמשים חיצוניים ובכך לגרום לשיבוש אצל המשתמשים המעוניינים להתחבר אל השרת.

**פירוט רכיבי המערכת**

**לקוח (Client)**

מודול client.py :



Class VerticalScrolledFrame(tki.Frame)

1. **פונקציית \_\_init\_\_:**

הפונקציה הבונה על המחלקה המגדירה את בר הגלילה של מסך חדר הצ'אט. הפונקציה מאתחלת את המחלקה במשתנה הבא:

*self.interior –* Frame אשר נמצא בתוך הCanvas אשר בתוכו יש גלילה

טענת כניסה: מקבלת self.root

טענת יציאה: יוצרת בר גלילה לFrame אשר נמצא בתוך ה-Canvas בגודל הרצוי.

1. **פונקציית \_configure\_interior:**

פונקציה העוקבת אחר השינויים ברוחב של ה-Frame הפנימי וה-Canvas ומסנכרנת ביניהם בהתאם ומעדכנת את הScrollbar.

טענת כניסה: מקבלת event

טענת יציאה: מעדכנת את ה-Scrollbar וה-Canvas על מנת שגודלם יהיה מתאים לFrame הפנימי ובכך יעבדו בסנכרון.

1. **פונקציית \_configure\_canvas:**

פונקציה המעדכנת את הרוחב של ה-Frame הפנימי על מנת שתוכל למלא את ה-Canvas.

טענת כניסה: מקבלת event

טענת יציאה: מעדכנת את הרוחב של ה-Frame הפנימי על מנת שתוכל למלא את ה-Canvas.

Class EMessageClient:

1. **פונקציית \_\_init\_\_:**

הפונקציה הבונה מגדירה את התקשורת העתידית אשר אמורה להיווצר בין הלקוח לשרת. הפונקציה מאותחלת על-ידי המשתנה הבא:

*self.socket –* משתנה המגדיר את התקשורת בין הלקוח לשרת.

טענת כניסה: אין

טענת יציאה: מגדירה את התקשורת בין הלקוח לשרת.

1. **פונקציית connect\_to\_server(self):**

פונקציה המנסה להתחבר לשרת, המתבקש על-ידי הלקוח, ופועלת בהתאם אם מצליחה או שאינה מצליחה.

טענת כניסה: אין

טענת יציאה: מבקשת חיבור לשרת ופועלת בהתאם לתשובת השרת.

1. **פונקציית send\_message(self, message):**

פונקציה המקבלת הודעה ושולחת אותה אל השרת.

טענת כניסה: message

טענת יציאה: שולחת את ההודעה לשרת.

1. **פונקציית recv(self, size=1024):**

פונקציה המנסה לקבל הודעות אשר נשלחות מן השרת אשר התקבלו אליו מן המשתמש השני, בגודל תווים שאינו קבוע.

טענת כניסה: size=1024

טענת יציאה: מקבלת הודעה מהשרת.

1. **פונקציית close\_server\_connection(self):**

פונקציה הסוגרת את החיבור בין הלקוח לשרת.

טענת כניסה: אין

טענת יציאה: סוגרת את החיבור של הלקוח עם השרת.

Class EMessageGUI:

1. **פונקציית \_\_init\_\_:**

הפונקציה הבונה מגדירה את המשתנים אשר בעזרתם נבנית הקונפיגורציה (העיצוב וההגדרות) של המוצר. הפונקציה מאותחלת על-ידי המשתנים הבאים:

*self.root()* – משתנה המגדיר את ה-root, הבסיס של מסך המוצר.

*self.eclient() –* משתנה מסוג EMessageClient האחראי על החיבור והתקשורת בין שני הלקוחות

*self.menubar* – משתנה האחראי על התפריט אשר מוצג בחלק העליון של המסך במהלך השימוש במוצר

*self.white\_box* – משתנה מסוג תמונה, המייצגת את בועת הדיבור של המשתמש הנגדי. מוצגת כאשר המשתמש מקבל הודעת טקסט אשר נשלחת מהמשתמש הנגדי.

*self.green\_box -* משתנה מסוג תמונה, המייצגת את בועת הדיבור של הנגדי. מוצגת כאשר המשתמש מעוניין לשלוח הודעת טקסט למשתמש השני.

*self.txt\_input* – משתנה מסוג Entry אשר מכיל בתוכו את המידע אשר המשתמש מזין ומעוניין לשלוח אל המשתמש השני.

*self.authenticated* – משתנה בוליאני הבודק האם החיבור אשר התבצע בין הלקוח לשרת הצליח. במידה וכן יפתח חדר הצ'אט במקום מסך ההתחברות.

*self.welcome\_label –* משתנה מסוג Label אשר מציג הודעת פתיחה ללקוח אשר מתחבר לצ'אט.

*self.password* – משתנה המכיל את הסיסמה אשר הלקוח מזין במסך ההתחברות.

*self.error\_label* – משתנה מסוג Label המציג הודעת שגיאה של הלקוח במסך ההתחברות במידה והזין סיסמה שגויה או ששם המשתמש אשר הזין תפוס.

*self.username* – משתנה המכיל את שם המשתמש אשר הלקוח מזין במסך ההתחברות.

*self.content\_frame* – משתנה מסוג VerticalScrolledFrame(self.root), האחראי על גלילת ה-Scrollbar באופן האידיאלי.

*self.attempts* – משתנה המונה ומציג בתור self.error\_label את מספר הניסיונות שלוקחים ללקוח להתחבר לחדר הצ'אט כשורה.

*self*.*username\_input* – תיבת הטקסט במסך הפתיחה המקבלת שם משתמש מהלקוח

*self*.*password\_input –* תיבת הטקסט במסך הפתיחה המקבלת סיסמה מהלקוח

*self*.*emojies\_bar* – משתנה מסוג Frame אשר בו ממוקמים כל הכפתורים אשר מייצגים את האימוג'ים הזמינים ללקוח לשימוש

*self*.*scroller* – משתנה מסוג Scrollbar האחראי על הגלילה של הצ'אט

*self*.*other\_username\_label –* משתנה מסוג Label אשר מציג את שם המשתמש הנגדי במידה והוא מחובר על המסך.

טענת כניסה: אין

טענת יציאה: מגדירה את המשתנים הכלליים המייצגים את הקונפיגורציה (העיצוב וההגדרות) של המוצר.

1. **פונקציית configure\_login\_window(self):**

הפונקציה בונה כרצוי את כלל המאפיינים והקונפיגורציה (העיצוב וההגדרות) אשר מוצגים / אמורים להיות מוצגים במקרים מסוימים על פני מסך ההתחברות.

טענת כניסה: אין

טענת יציאה: מגדירה את כלל המאפיינים והקונפיגורציה (העיצוב וההגדרות) אשר מוצגים על פני מסך ההתחברות.

1. **פונקציית login\_func(self, event=None):**

הפונקציה בודקת האם הפרטים אשר המשתמש הזין מתאימים (סיסמה נכונה, שם משתמש שהינו זמין), במידה וכן יכנס לחדר הצ'אט, במידה ולא – תתקבל הודעה ששם המשתמש או סיסמתו אינם נכונים ושעליו לנסות להזין בשנית.

טענת כניסה: event=None

טענת יציאה: בודקת האם הפרטים אשר המשתמש הזין נכונים על מנת להתחבר או שאינם (סיסמה נכונה, שם משתמש שהינו זמין) ופועלת בהתאם.

1. **פונקציית exit\_func(self):**

פונקציה הנכנסת לפעולה כאשר המשתמש לוחץ על הכפתור X במסך ההתחברות, ומציגה הודעה קופצת אשר שואלת אותו האם הינו בטוח שברצונו לצאת. במידה וילחץ Yes, המוצר יסגר לחלוטין. במידה וילחץ No, המוצר יפעל כרגיל.

טענת כניסה: אין

טענת יציאה: בודקת האם המשתמש בטוח שברצונו לצאת מן המוצר ופועלת בהתאם להנחייתו.

1. **פונקציית start\_emessage(self):**

פונקציה המכילה Thread הפועל על הפונקצית get\_message(self), הבודקת באופן תמידי האם מתקבלת הודעה מהמשתמש הנגדי או נשלחת הודעה מהמשתמש.

טענת כניסה: אין

טענת יציאה: מפעילה Thread האחראי על בקרת קבלת ההודעות.

1. **פונקציית quit(self):**

פונקציה הפועלת כאשר המשתמש נמצא במסך חדר הצ'אט, כאשר הוא לוחץ על כפתור ה-X, הפונקציה תשבית את פעילות הלקוח לחלוטין ללא כל שגיאות.

טענת כניסה: אין

טענת יציאה: סוגרת את פעילות המוצר לחלוטין.

1. **פונקציית configure\_chat\_window(self):**

הפונקציה בונה כרצוי את כלל המאפיינים והקונפיגורציה (העיצוב וההגדרות) אשר מוצגים / אמורים להיות מוצגים במקרים מסוימים על פני מסך חדר הצ'אט

טענת כניסה: אין

טענת יציאה: מגדירה את כלל המאפיינים והקונפיגורציה (העיצוב וההגדרות) אשר מוצגים על פני מסך חדר הצ'אט.

1. **פונקציית fill\_emojies(self):**

הפונקציה מעלה ומציגה בתור כפתורים את כלל התמונות הנמצאות בתיקייה מסוימת המוגדרת בתוך הפונקציה (במקרה זה, תיקייה המכילה תמונות של אימוג'ים מפורמט PNG). כאשר המשתמש ילחץ על אחד מהכפתורים אשר כל אחד מהם מכיל אימוג'י (פרצוף, ציור, הבעת רגש) שונה, הוא ישלח את אותו האימוג'י למשתמש הנגדי.

טענת כניסה: אין

טענת יציאה: מעלה ומציגה בתור כפתורים את האימוג'ים הרצויים.

1. **פונקציית upload\_image(self):**

פונקציה הנכנסת לפעולה כאשר המשתמש לוחץ על כפתור העלאת התמונה. הפונקציה מעלה תמונה, מציגה אותה, ממירה את הdata שלה למחרוזת מסוג base64 על פני מסך חדר הצ'אט וקוראת לפונקציה send\_image(self, photo, resize=200).

טענת כניסה: אין

טענת יציאה: מעלה תמונה, מציגה אותה וקוראת לפונקציה נוספת.

1. **פונקציית send\_image(self, photo, resize=200):**

פונקציה המקבלת מחרוזת מסוג base64 של תמונה אשר הועלתה דרך הפונקציה upload\_image(self). הפונקציה שולחת לשרת את: שם התמונה, אורך התמונה, הגודל הרצוי שלה, והמידע מסוג base64.

טענת כניסה: photo, resize=200

טענת יציאה: שולחת את שם התמונה, אורכה, גודלה הרצוי והמידע שלה מסוג base64.

1. **פונקציית send\_message(self, event=None):**

פונקציה השולחת את הודעת הטקסט אשר המשתמש מזין אל המשתמש הנגדי במידה ואורכה הוא בין תו אחד ל-24 תווים.

טענת כניסה: event=None

טענת יציאה: שולחת הודעת טקסט לשרת.

1. **פונקציית get\_message(self):**

פונקציה האחראית על קבלת ההודעות מן השרת, על בקרת קבלת ההודעות, כלומר: האם ההודעה היא הודעת תמונה או הודעת טקסט.

טענת כניסה: אין

טענת יציאה: מציגה למסך את הודעת הטקסט/התמונה אשר התקבלה על-ידי המשתמש הנגדי או שנשלחה על-ידי המשתמש.

1. **פונקציית my\_message(self, message):**

פונקציה האחראית על בקרת קבלת ההודעות, כלומר: הפונקציה בודקת האם ההודעה האחרונה שנשלחה על-ידי השרת במקור נשלחה מהמשתמש שלי או מהמשתמש הנגדי ופועלת בהתאם בפונקצייה get\_message(self):

טענת כניסה: message (הודעה).

טענת יציאה: מחזירה True/False בהתאם לאם ההודעה האחרונה שנשלחה היא על ידי או על-ידי המשתמש הנגדי.

פונקציות ראשיות בClient.py

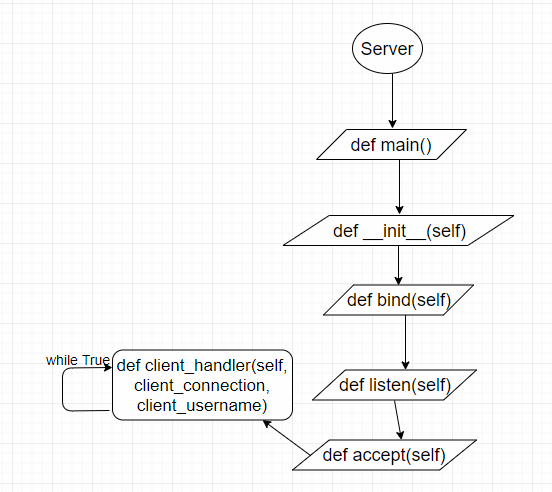
1. **פונקציית main():**

פונקציה המכילה משתנה מסוג EMessageGUI(). הפונקציה קוראת ל-configure\_login\_window() ומריצה את המשתנה באמצעות mainloop().

טענת כניסה: אין

טענת יציאה: מריצה את המשתנה וקוראית לפונקציה בתור המחלקה הבונה את המוצר.

**שרת (Server)**

מודול server.py :

Class EMessageServer:

1. **פונקציית \_\_init\_\_:**

הפונקציה הבונה מגדירה את התקשורת העתידית אשר אמורה להיווצר בין השרת ללקוח. הפונקציה מאותחלת על-ידי המשתנה הבא:

*self.socket –* משתנה המגדיר את התקשורת בין השרת ללקוח.

*self.clients\_list –* משתנה מסוג רשימה המכיל את שמות הלקוחות המחוברים לשרת.

טענת כניסה: אין

טענת יציאה: מגדירה את התקשורת בין הלקוח לשרת ורשימת לקוחות מחוברים לשרת.

1. **פונקציית bind(self):**

הפונקציה מגדירה את הדרישות אשר דרושות ללקוח להתחבר לשרת (IP של השרת וPort).

טענת כניסה: אין

טענת יציאה: מגדירה את הדרישות אשר דרושות ללקוח להתחבר לשרת.

1. **פונקציית listen(self):**

הפונקציה מאזינה לחיבורים מלקוחות אשר רוצים להתחבר לשרת. זאת כי פונקציה זו נדרשת למען קבלת חיבורים מהלקוחות.

טענת כניסה: אין

טענת יציאה: מאזינה לחיבורים מלקוחות שרוצים להתחבר לשרת.

1. **פונקציית accept(self):**

הפונקציה מקבלת חיבורים ללקוחות חדשים, את שם המשתמש והסיסמה אשר הוזנו על-ידי הלקוח וקוראת לפונקציה הבודקת האם שם המשתמש זמין והאם הסיסמה נכונה. במידה ופרטי הלקוח אשר הזין נכונים/זמינים, יכניס השרת את שמו לרשימת שמות המשתמשים, ולרשימה נוספת המציינת את הלקוחות המחוברים.

בנוסף הפונקציה מפעילה Thread על הפונקציה client\_handler אשר בודקת באופן תמידי האם נשלחה הודעה מאחד הלקוחות.

טענת כניסה: אין

טענת יציאה: מקבלת חיבור חדש מלקוחות, ומפעילה Thread אם פרטי הלקוח אשר הזין נכונים/זמינים על פונקציה המקבלת הודעות מהלקוחות.

1. **פונקציית client\_handler(self, client\_connection, client\_username):**

פונקציה המקבלת חיבור של לקוח לשרת ואת שם המשתמש אשר הלקוח הזין.

הפונקציה שולחת הודעה לכלל השרתים המכילה את שמות המשתמשים של הלקוחות המחוברים, על מנת שפונקציה בלקוח תוכל לבצע את בקרת ההודעות: מי שלח את ההודעה.

בנוסף, הפונקציה בודקת באמצעות הודעה הנשלחת מן הלקוח האם ההודעה אשר נשלחה היא תמונה/אימוג'י או שהיא הודעת טקסט ומטפלת בה בהתאם.

טענת כניסה: client\_connection, client\_username (שם משתמש וחיבור)

טענת יציאה: מבצעת בקרה לסוג התמונה אשר נשלחה על-ידי אחד הלקוחות.

1. **פונקציית verify(self, username, password):**

פונקציה המקבלת שם משתמש וסיסמה ובודקת ומחזירה True/False בהתאם לאם הפרטים אשר הלקוח הזין במסך ההתחברות נכונים/זמינים או לא.

טענת כניסה: username, password (שם משתמש וסיסמה).

טענת יציאה: מחזירה True/False בהתאם לנכונות/זמינות של פרטי הלקוח אשר הזין.

1. **פונקציית send\_all(self, message, without=None):**

פונקצית המקבלת הודעה, ו-without המציין למי מהלקוחות לא לשלוח את ההודעה, ושולחת את ההודעה אשר נשלחה מאחד הלקוחות לכלל הלקוחות ברשימה.

טענת כניסה: message, without=None (הודעה וwithout מציין למי מבין הלקוחות הקיימים לא לשלוח את ההודעה).

טענת יציאה: שולחת הודעה אשר התקבלה מאחד הלקוחות לכלל הלקוחות, אלא אם כן ישנו לקוח אשר נמצא ב-without ואליו לא תישלח ההודעה.

פונקציות ראשיות ב-Server.py:

1. **פונקציית main():**

פונקציה המכילה משתנה מסוג eMessageServer , קוראת לפונקציות השונות במחלקה ש: מגדירות את הדרישות לחיבור, מאזינות לחיבורים מהלקוח, מקבלות חיבורים חדשים.

טענת כניסה: אין

טענת יציאה: מריצה את הפונקציות השונות על מנת ביצוע חיבור מהימן.

**דיון בנושא העיצוב הנבחר**

את הפרוייקט בחרתי לתכנת בשפה Python, מכיוון שזוהי השפה נוחה לשימוש לפי דעתי, קל ללמוד אותה, היא שפה מאוד פופולארית כיום לתכנות ובעלת מודולים ופונקציות מובנות שהיה לי מראש את הידע בהן אשר נעזרתי בהן במהלך בניית המוצר.

בנוסף, את ממשק המשתמש יצרתי באמצעות הספרייה המובנית Tkinter, מכיוון שהיה לי בה ידע קודם, היא מאוד פופולארית, יש עליה הרבה מידע ברחבי האינטרנט, והיא תואמת לחלוטין את הצרכים והדרישות של הפרויקט שלי מבחינת הנראות.

את החלוקה למודולים בחרתי על-פי תחום עיסוק, כל מודול עוסק בתחום אחר, חלקם חשובים מאוד ותפקודם משמעותי ביותר, חלקם פחות ומטרתם על מנת לממש את נוחות השימוש והגישה הידידותית למשתמש של המוצר.

כך המודולים הופכים את התוכנה לקלה לתחזוקה.

**עיצוב נתונים ופרוטוקולים**

**Client (לקוח):**

הודעות הטקסט של הלקוח שאותן הוא רוצה לשלוח אל השרת, בתוך הפונקציה send\_message(self, event=None) במחלקה EMessageGUI ובנוסף כאשר הלקוח מעוניין לשלוח תמונה אל השרת, התמונה נשלחת על-ידי הפונקציה send\_image(self, photo, size=200).

**Server (שרת):**

הודעות הטקסט אשר נשלחות על-ידי הלקוח עוברות טיפול בפונקציה client\_handler(self, client\_connection, client\_username) ועוברות בקרה בנוגע לסוגן (טקסט/תמונה או אימוג'י).

**פרוטוקולים**

התקשורת נעשית באמצעות פרוטוקול תקשורת של לקוח-שרת באמצעות socket המבוסס על פרוטוקול TCP.

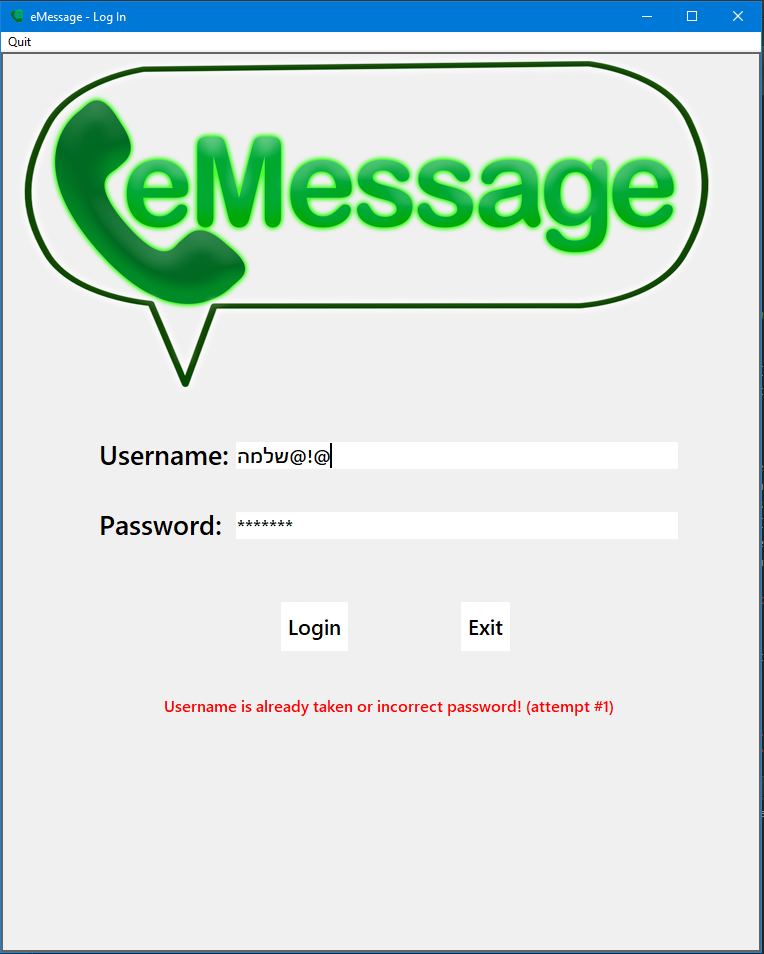
TCP היא חבילת פרוטוקולי תקשורת שעליה מושתתת רשת האינטרנט, ושם למודל שכבתי המתאר תקשורת ברשתות מחשבים. על שכבת הקשר של הפרוטוקול ניתן להסתכל כשתי שכבות שונות: אחת המקשרת באופן פיזי בין חולייה ברשת לחולייה אליה צריך למסור את המידע (link layer), ואחת שפיזית מבצעת את תמסורת הנתונים (physical layer).

**מסמך בדיקות**

1. **הרשמה למערכת**

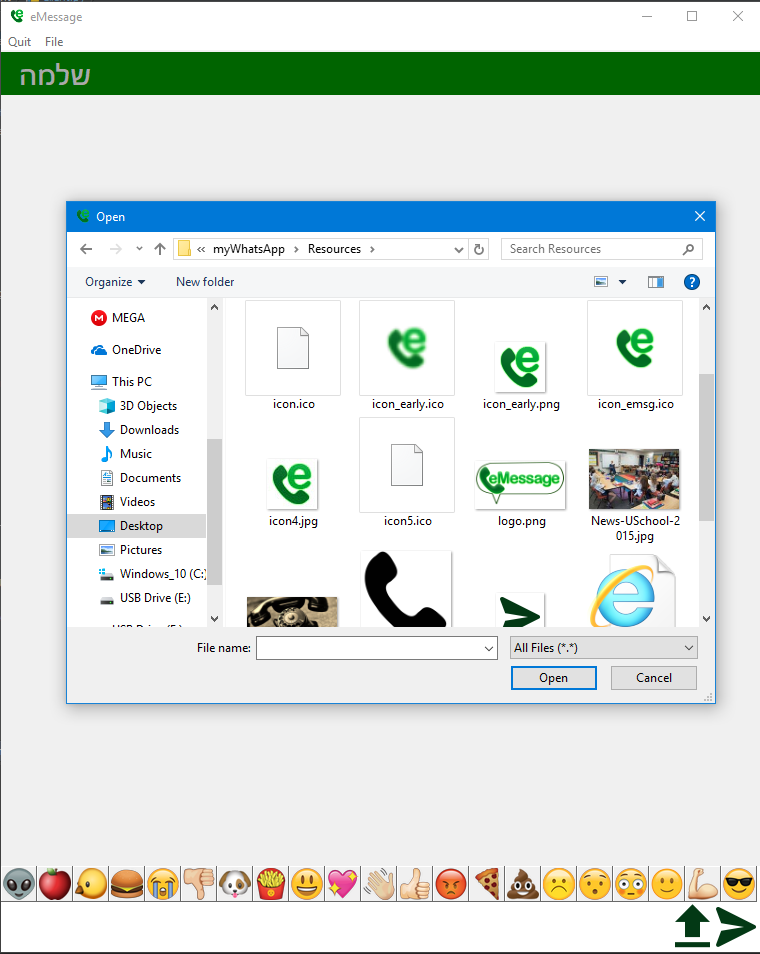
במקרה ומתקבלים שם משתמש ו/או סיסמה אשר אינם עומדים בקריטריונים הבאים:

* אסור ששם המשתמש יהיה מחרוזת ריקה.
* אסור שאורך שם המשתמש יעלה על 16 תווים.
* אסור ששם המשתמש יכיל תווים מיוחדים כגון: @#$%^&\*)(-+;
* על הסיסמה להיות הסיסמה אשר הוגדרה מראש בתוך השרת. סיסמה זו אמורה להיות ידועה מראש לכלל המשתמשים.



**2. יציאה מחלון בחירת התמונה**

במידה והמשתמש לוחץ על כפתור ה"**Upload"** המסומן על-ידי חץ כלפי מעלה, תופעל הפונקציה upload\_image(self) אשר פותחת חלון שבאמצעותו ניתן לבחור תמונה שאותה ניתן לשלוח למשתמש השני. אבל במידה המשתמש מתחרט ומחליט ללחוץ על כפתור ה-X, לא תופיע שום שגיאה.

ברמת הקוד, התוסף של העלאת התמונה נשמר בתוך משנה המקבל את המידע אודות התמונה. במידה ולא התקבל כלום, הייתה נוצרת הודעת שגיאה בקוד, אך נוצר תנאי שאומר כי אם המשתנה אשר אמור לקבל תמונה לא קיבל דבר, הפונקציה לא תחזיר דבר.

**מסמך מחקר**

לאחר שבחרתי את נושא הפרויקט, שמאוד הלהיט אותי ומאוד רציתי ליצור פרויקט בקונספט אשר תכננתי מראש, הדבר הראשון שהתחלתי לחקור היה בחירת הספרייה שבה אשתמש לצורך בניית ממשק המשתמש שלי. הממשק הראשון שעלה לי לראש הוא Tkinter, משום שיצא לי לעבוד עם Tkinter בכיתה י', בפרויקטים קודמים במגמת הסייבר שנדרש מאיתנו לעשות, ומאוד נהניתי להשתמש בספרייה זו ככה שאני הרגשתי שמבחינת הגרפיקה אני יכול להגיע לעיצוב שאני רוצה בזכות הפונקציות והתוספים שיש בתוך ספרייה זו.

היה מאתגר לבצע את העיצוב, את המעבר ממסך ההתחברות למסך הצ'אט, וגם את הבקרה על שם המשתמש והסיסמה או כל מיני תוספים המשפרים את ידידותיות התוכנה למשתמש או הופכים את המוצר ליותר נחמד ומעניין.

השלב שבו התקשיתי הכי הרבה, למרבה ההפתעה, היה החלק של ה-Scrollbar, שלצערי נאלצתי ליצור מחלקה מיוחדת בשבילו על מנת לממש אותו בצורה תקינה.

מראש היה לי הרצון שיהיה שרת שאליו יתחברו לקוחות שונים ושכל אחד מהלקוחות יתופעל באופן זהה לשני.

בנוסף קושי שני שהיה לי הייתה קבלת המידע אשר נשלח בזמן אמת ובאופן שלאו דווקא הדדי. כלומר, המטרה שלי הייתה שהתקשורת לא תהיה מעבר של הודעה ממשתמש אחד, הממתין להודעה ממשתמש שני, שני המשתמשים אינם מוגבלים במספר ההודעות שהם יכולים לשלוח ברצף ללא הודעה אשר נשלחת מהמשתמש השני.

לשם כך השתמשתי ב-Thread, אחד בתוך המחלקה של השרת והשני בתוך המחלקה של הלקוח בהתאם. הThread של השרת אשר מתמקד על הפונקציה בתוך המחלקה שבודקת אם התקבל מידע חדש ומאיזה סוג , ואילו ה-Thread של הלקוח מתמקד על הפונקציה המקבלת הודעות מן השרת ומבצעת בקרה על מנת לדעת את סוג ההודעה (טקסט/תמונה או אימוג'י) ומאין ההודעה הגיעה מלכתחילה (ממני או מהמשתמש הנגדי).

**מדריך למשתמש**

לקוח (Client):

דרישות מערכת:

* מערכת הפעלה של Windows XP ומעלה
* Python 2.7 ומעלה.
* חיבור לרשת

**מדריך שימוש – לקוח (Client)**

1. יש להריץ את קובץ הפייתון של הלקוח (בידיעה שקובץ הפייתון של השרת כבר מופעל/נפתח), לאחר מכן יפתח בפניכם המסך הבא.

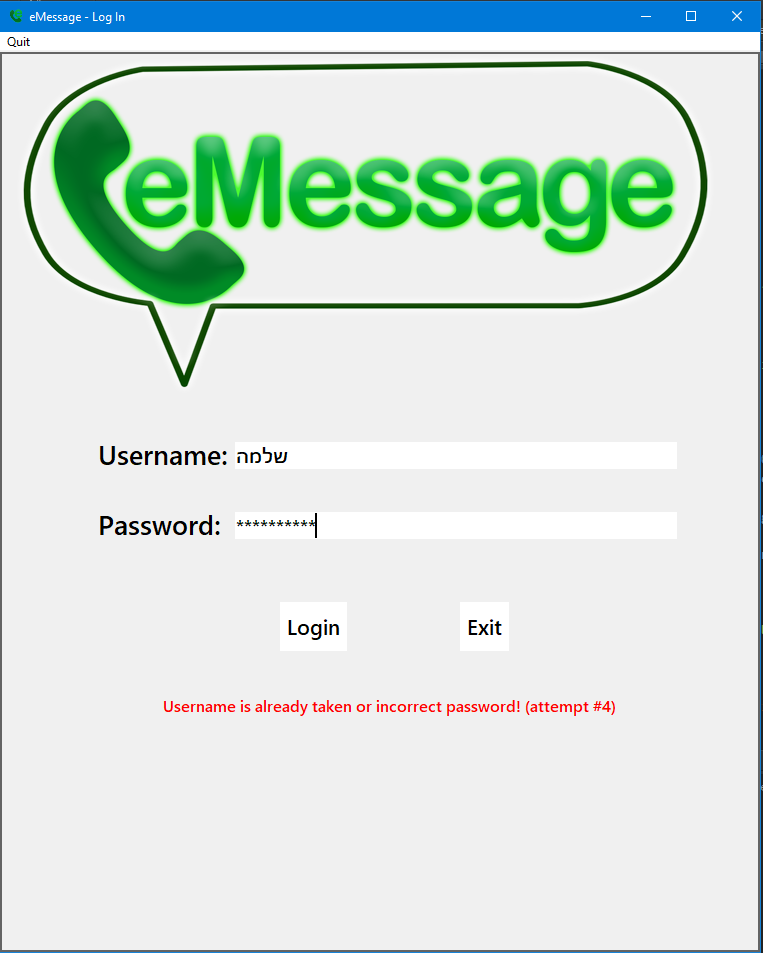


המסך שנפתח בפניכם נקרא **"מסך ההתחברות"**.

כפי שניתן לראות החיבור לצ'אט נעשה באמצעות הזנת שם משתמש וסיסמה.

עליכם להזין את שם המשתמש, כל שם אשר תרצו אלא אם כן הוא ריק, ארוך מידי (מעל ל-16 תווים) ומכיל סמלים מיוחדים. לאחר מכן, עליכם להזין את הסיסמה הנכונה, שהיא **123**.

במידה ותזינו שם משתמש או סיסמה שאינם עומדים בתנאים הבאים, תקבלו הודעת שגיאה **אדומה** אשר מציינת כי שם המשתמש אינו זמין או שהסיסמה אינה נכונה,

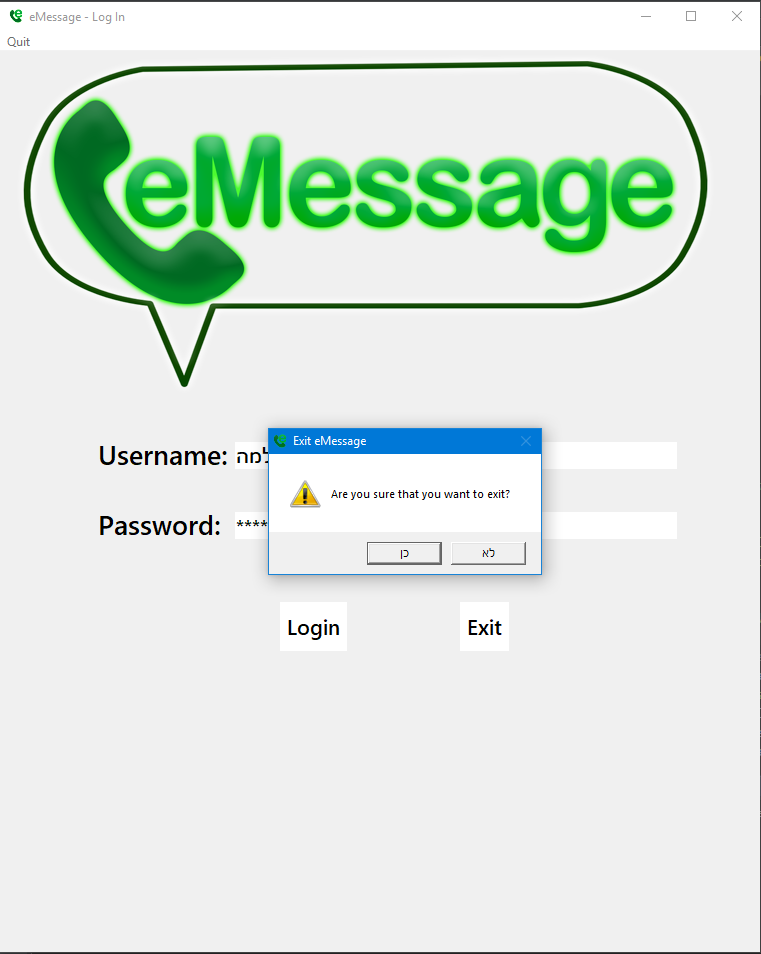


במידה ותרצו לצאת ממסך ההתחברות, באמצעות לחיצה על כפתור ה-"**Exit"** או כפתור ה-**X,** תופיע בפניכם הודעת שאלה השואלת אתכם האם אתם בטוחים שאתם רוצים לצאת.

המטרה של כפתור זה היא שבמידה ובטעות לחצתם על כפתור היציאה, שהצ'אט לא ייסגר לגמרי וששני המשתמשים שוב יצטרכו להתחבר מחדש.

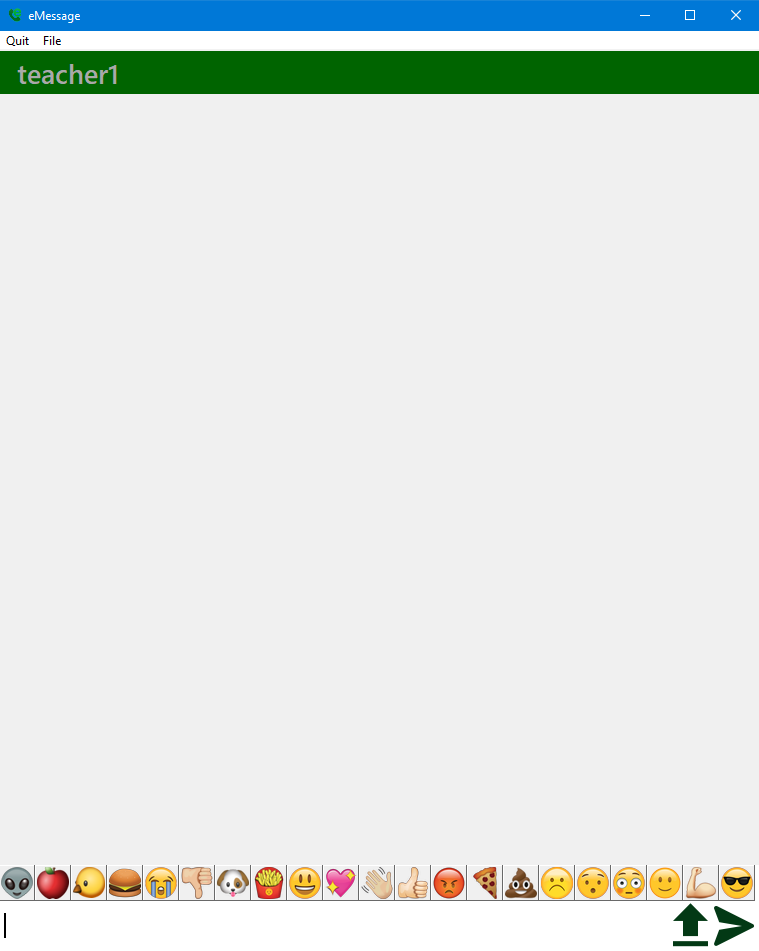
במידה ותלחצו על "**לא**", הצ'אט ימשיך מאותה הנקודה אשר היה בה, במידה ותלחצו על "**כן**" מסך הצ'אט ייסגר לחלוטין.

במידה ותרצו להתחבר מחדש לצ'אט, תצטרכו לעבור שנית דרך מסך ההתחברות.



1. כאשר הזנתם שם משתמש זמין ואת הסיסמה הנכונה אשר צוינה לעיל, יפתח בפניכם מסך חדר הצ'אט.

6



7

5

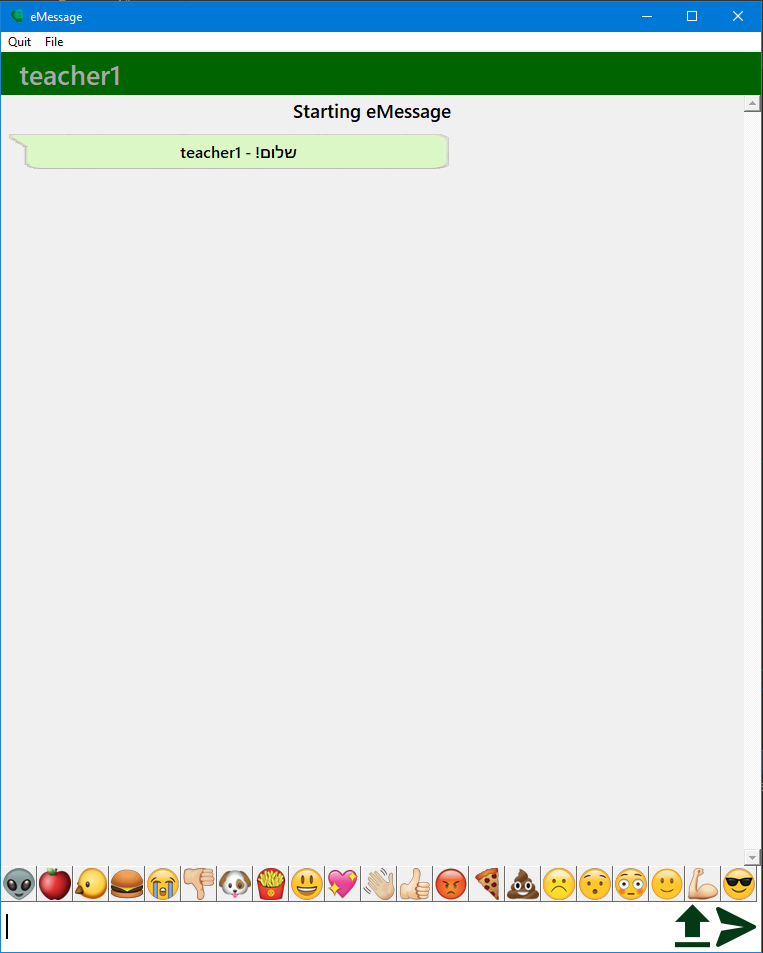
4

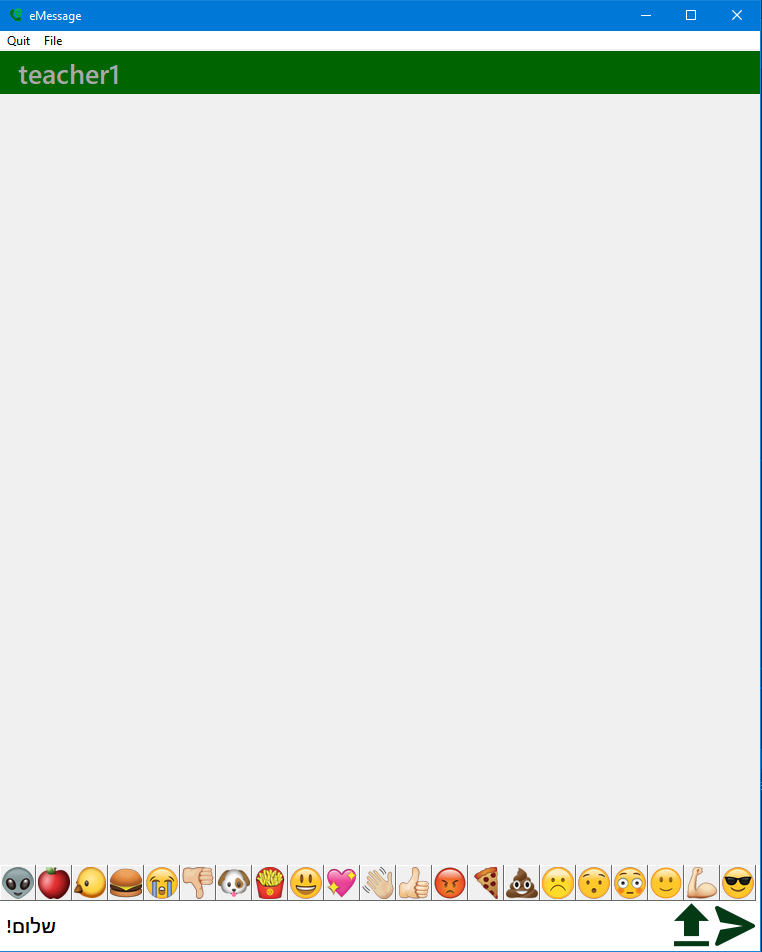
3

2

1

מסך חדר הצ'אט מכיל את הדברים הבאים:

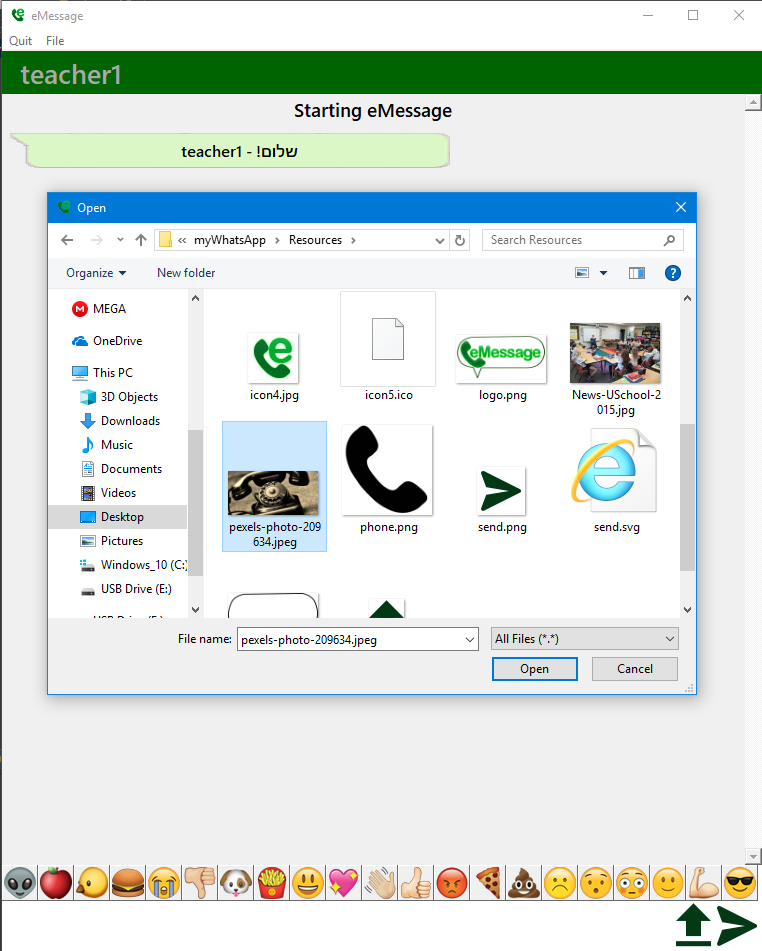
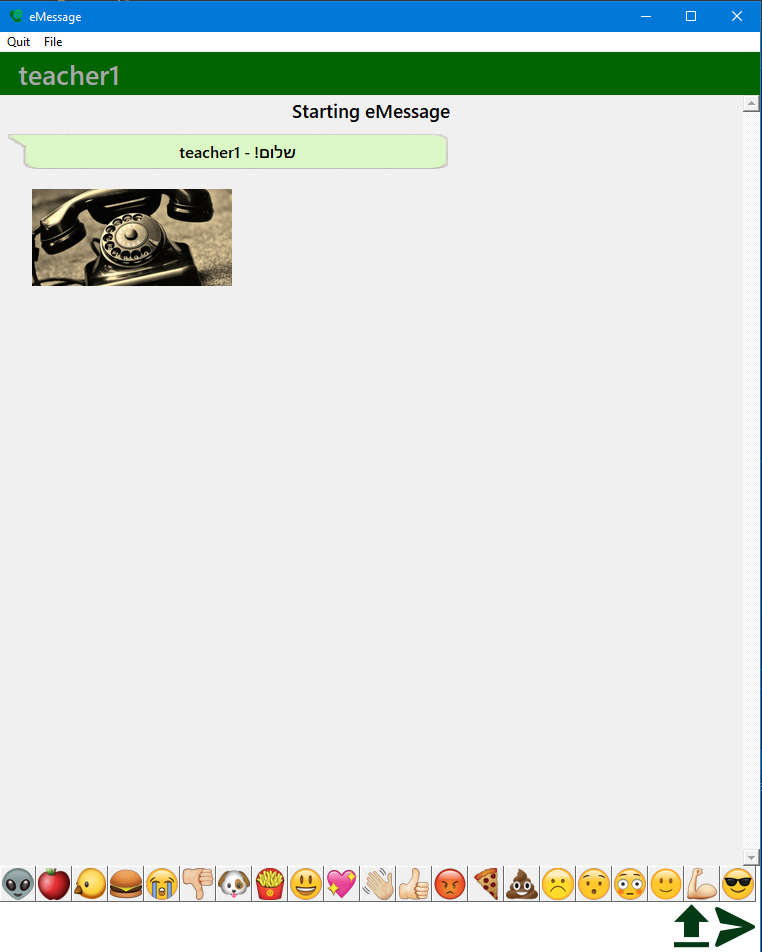
1. **כפתור שליחה**- משמש למען שליחת הודעות טקסט, מציג את ההודעה על המסך. בנוסף ניתן לשלוח הודעה על-ידי לחיצה על מקש ה-Enter במקלדת.



2

1

1. **כפתור העלאת התמונה-** באמצעות לחיצה על כפתור זה, יפתח בפניכם חלון שבו תוכלו לבחור תמונה אחת כלשהי. כאשר תלחצועל התמונה היא אוטומטית תופיע לכם על המסך ותישלח למשתמש השני.

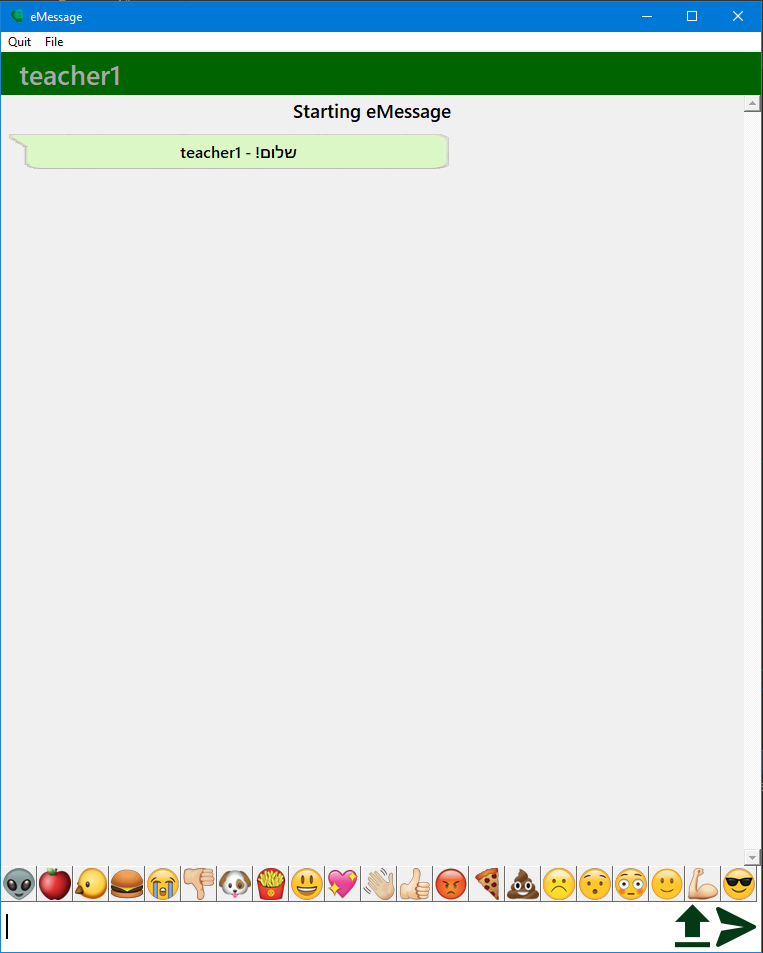
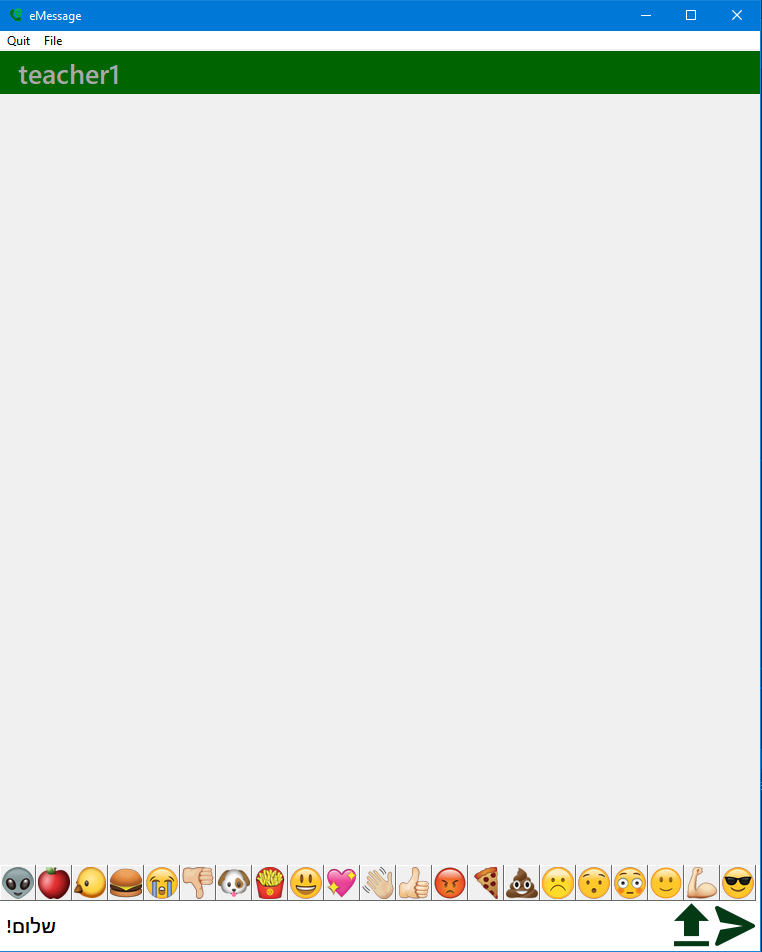


2

1

1. **שדה כתיבת הודעה-**

בשדה זה תוכלו לכתוב כל הודעת טקסט שבא לכם מלבד הודעה ריקה או הודעה שאורכה עולה על כ-30 תווים, וכאשר תלחצו על מקש ה-Enter על המקלדת, ההודעה תוצג לכם על המסך ותישלח למשתמש השני.



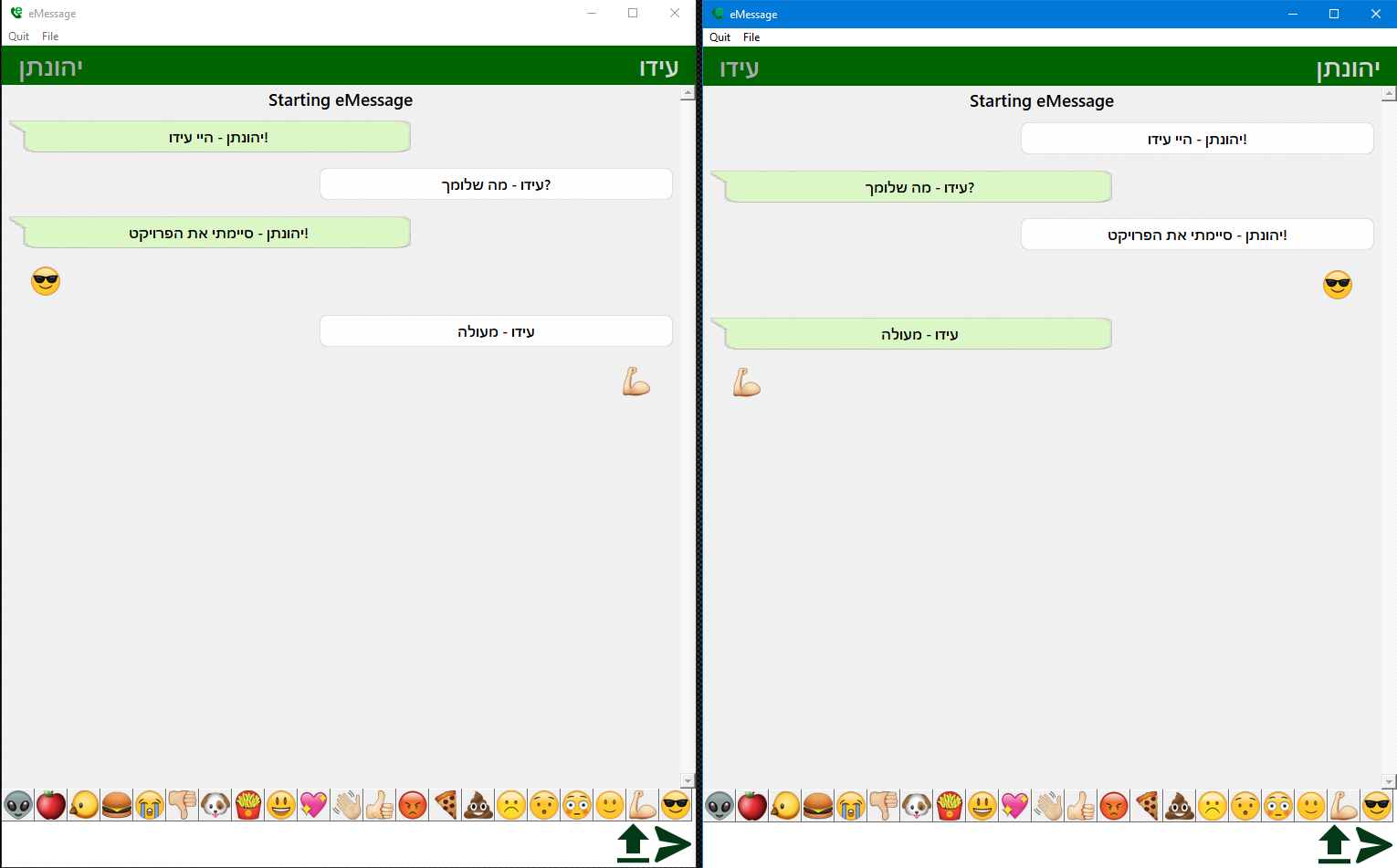
2

1

2

1. **סרגל האימוג'ים (Emojis)**

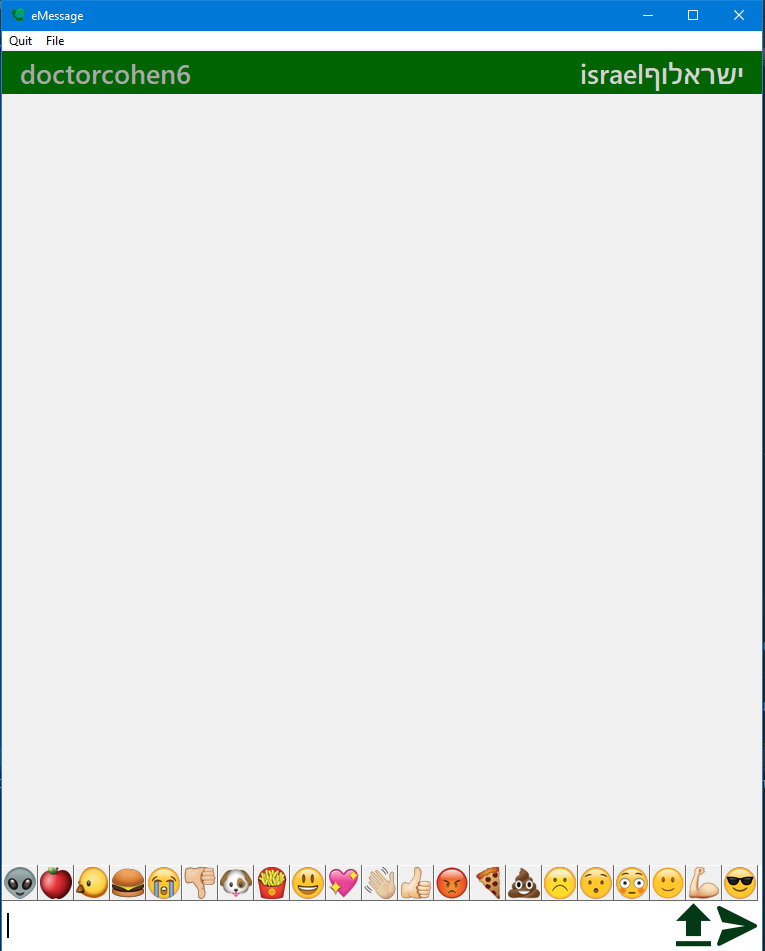
סרגל זה מכיל מספר רב של אימוג'ים – פרצופים או סמלים המביעים רגש או תחושה מסוימת. כאשר לוחצים על אחד מן האימוג'ים הבאים הוא נשלח אוטומטית בתור הודעה.





1. **סרגל שמות המשתמשים**

הסרגל בצבע ירוק, המכיל מצד שמאל את שם המשתמש אשר הוזן על-ידיך. כאשר יתחבר לצ'אט משתמש שני, יופיע בצד ימין על אותו הסרגל שם המשתמש שלו ברגע שהמשתמש השני יתחבר.



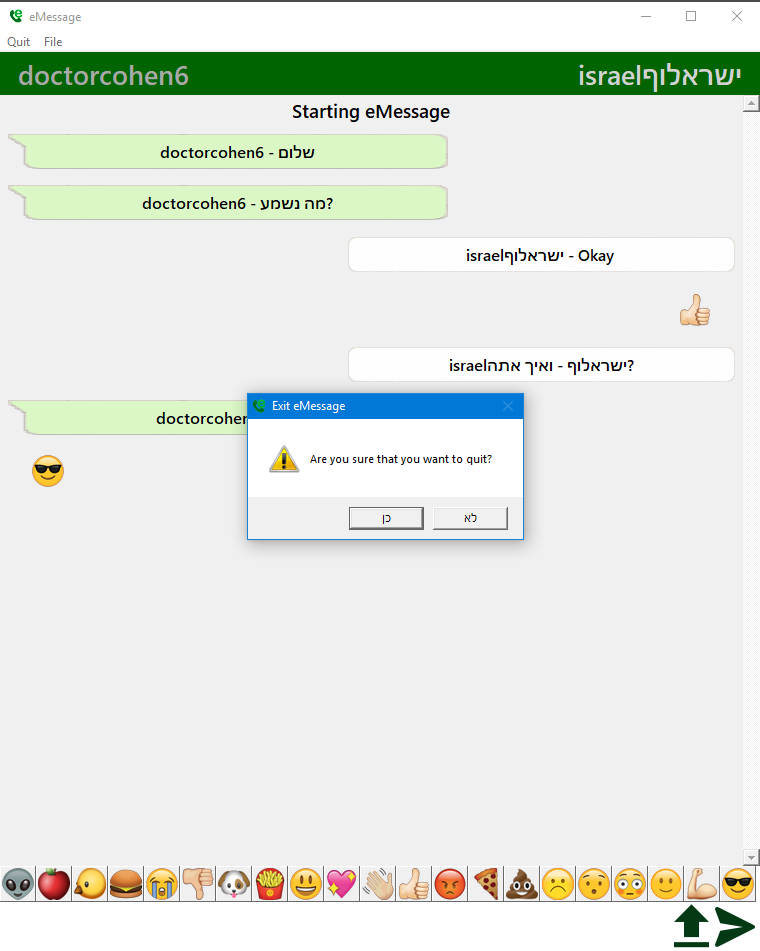
1. **כפתור ה-X – היציאה**

כאשר תלחצו עם העכבר על כפתור ה-X, תופיע בפניכם הודעת שאלה השואלת אתכם האם אתם בטוחים שאתם רוצים לצאת.

המטרה של כפתור זה היא שבמידה ובטעות לחצתם על כפתור היציאה, שהצ'אט לא ייסגר לגמרי וששני המשתמשים שוב יצטרכו להתחבר מחדש.

במידה ותלחצו על "**לא**", הצ'אט ימשיך מאותה הנקודה אשר היה בה, במידה ותלחצו על "**כן**" מסך הצ'אט ייסגר לחלוטין.

במידה ותרצו להתחבר מחדש לצ'אט, תצטרכו לעבור שנית דרך מסך ההתחברות.



1. **תפריט האפשרויות –**

נמצא בחלקו העליון של מסך חדר הצ'אט ומסך ההתחברות.

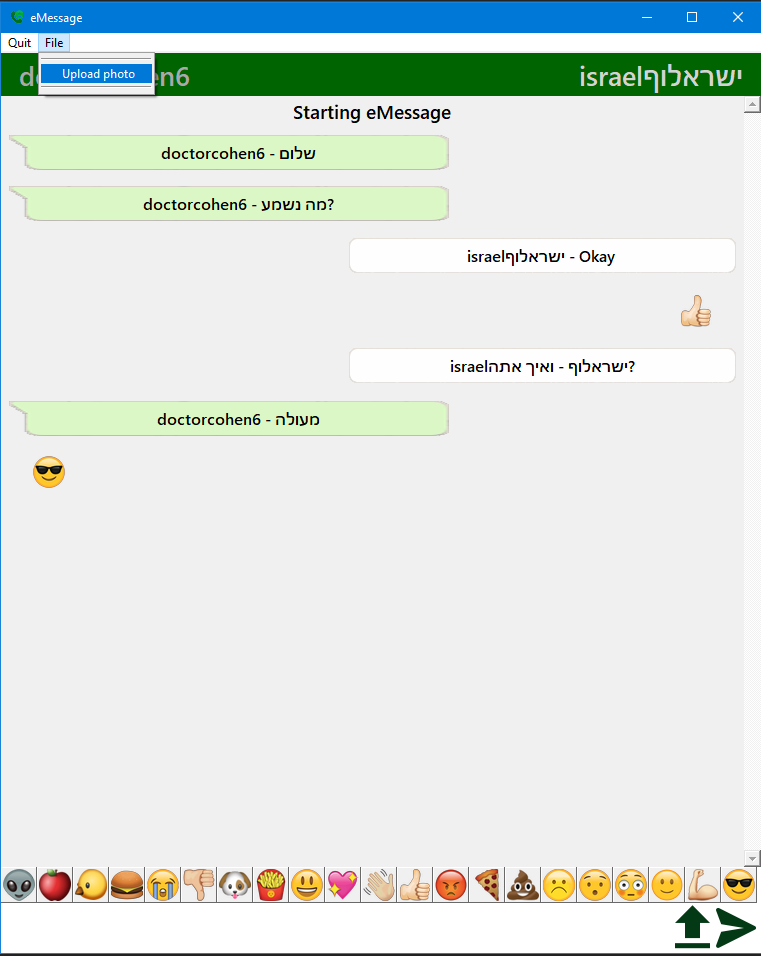
במסך ההתחברות תהיה אופציה אחת לשימוש בו הנקראת "Quit".

כאשר תלחצו על "Quit" תקפוץ בפניכם הודעה השואלת אתכם אם אתם בטוחים שברצונכם לצאת מהצ'אט. במידה ותלחצו על "**לא**", הצ'אט ימשיך מאותה הנקודה אשר היה בה, במידה ותלחצו על "**כן**" מסך הצ'אט ייסגר לחלוטין.

במידה ותרצו להתחבר מחדש לצ'אט, תצטרכו לעבור שנית דרך מסך ההתחברות.

מסך חדר הצ'אט מכיל את אותה האופציה ובנוסף מכיל את האופציה File, שכאשר לוחצים עליה ישנה תת אופציה הנקראת "Upload photo".

כאשר תלחצו על “Upload photo” , יפתח בפניכם חלון שבו תוכלו לבחור תמונה אחת כלשהי. כאשר תלחצועל התמונה היא אוטומטית תופיע לכם על המסך ותישלח למשתמש השני.



שרת (Server):

דרישות מערכת:

* מערכת הפעלה של Windows XP ומעלה
* Python 2.7 ומעלה.
* חיבור לרשת

**מדריך שימוש – לקוח (Client)**

* הפעולה היחידה אשר נדרשת מהמשתמש היא בסך הכל להפעיל את הקובץ, והוא יכנס לפעולה בעצמו ללא שום צורך בביצוע פעולה נוספת על-ידיכם (המשתמשים).