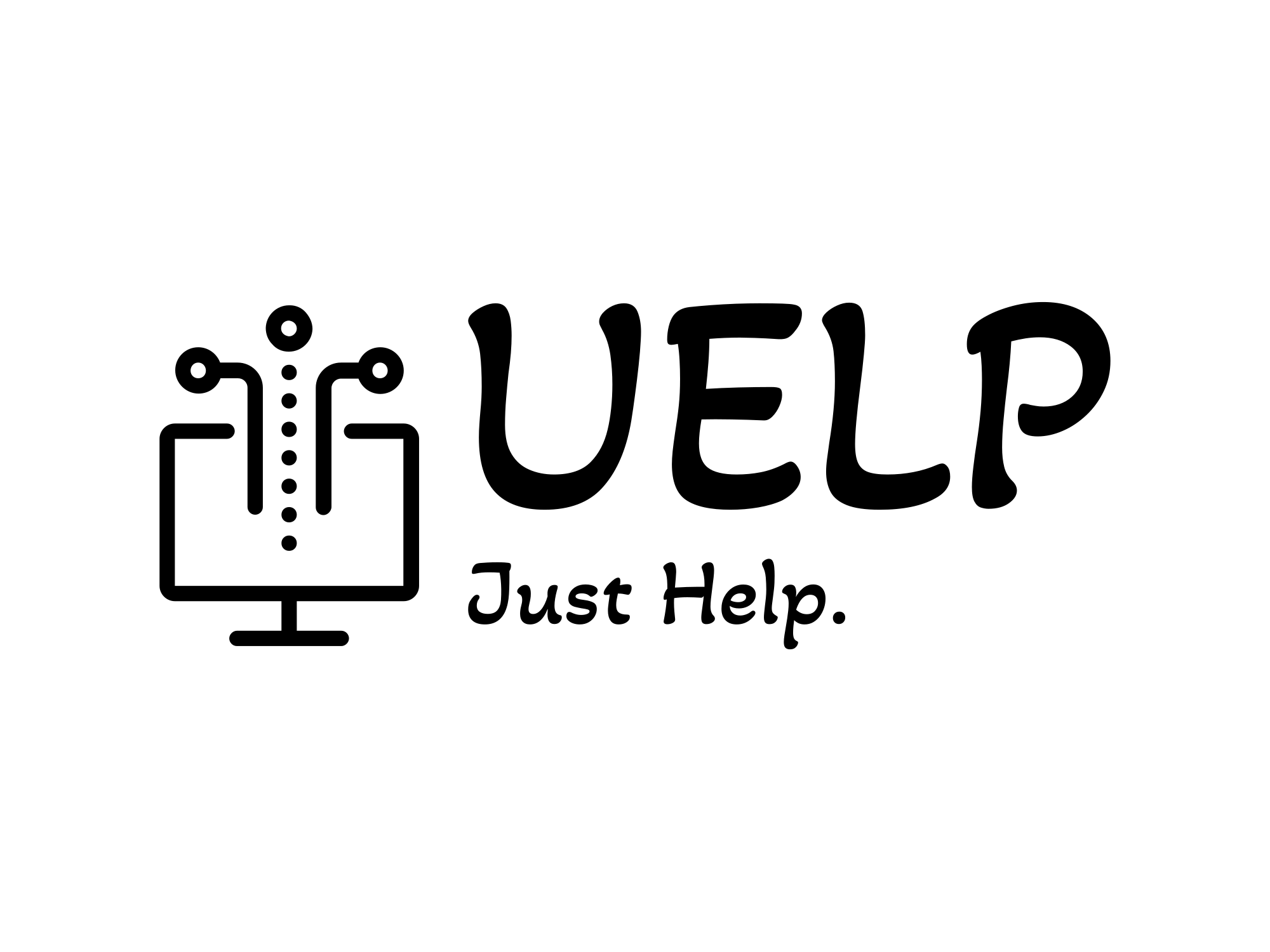
**תיק פרויקט- UELP**

**מערכת לבקרה ושליטה על מחשבים מרחוק**

**בית ספר: כרמים בנימינה**

**תלמיד: איתמר נחום**

**ת"ז תלמיד: 215871443**

**המנחה: מרי גבע**

**שם החלופה: מערכות הגנת סייבר**

**תאריך הגשה: 24/05/18**

תוכן עניינים

[פרק א' - ייזום 3](#_Toc166808873)

[פרק ב' - אפיון 6](#_Toc166808874)

[פרק ג' - ניתוח 13](#_Toc166808875)

[פרק ד' - העיצוב 21](#_Toc166808876)

[תיאור הארכיטקטורה 21](#_Toc166808877)

[תיאור הטכנולוגיה 21](#_Toc166808878)

[המודולים בשימוש 23](#_Toc166808879)

[סביבת הפיתוח 55](#_Toc166808880)

[האלגוריתמים המרכזיים 55](#_Toc166808881)

[מסכי הפרויקט 59](#_Toc166808882)

[פרוטוקול התקשורת 63](#_Toc166808883)

[מבני הנתונים 65](#_Toc166808884)

[סקירת חולשות והאיומים 66](#_Toc166808885)

[פרק ה' - הקוד 68](#_Toc166808886)

[פרק ו' - בדיקות 89](#_Toc166808887)

[פרק ז' - מדריך למשתמש 95](#_Toc166808888)

[פרק ח ' - רפלקציה 101](#_Toc166808889)

[פרק ט' - ביבליוגרפיה 102](#_Toc166808890)

[הקוד המלא- 103](#_Toc166808891)

# פרק א – UELP - ייזום

**תיאור ראשוני של המערכת-**

1. תיאור ראשוני של המערכת

**מערכת בקרת מחשב מרחוק היא תוכנה המאפשרת לאדם לסייע לאדם אחר ממחשב מרוחק.  
המערכת תחבר בין** Helper **(אנשים שמבינים במחשבים ומעוניינים לסייע לאחרים) לבין** AssistanceSeeker **(אנשים המעוניינים לקבל סיוע מאחרים)**

**המערכת תבטיח קבלת סיוע אך ורק מ-** Helper**-ים 'מאושרים' כדי שלא יגרם נזק ל-** AssistanceSeeker.

**המוצר הסופי אמור לאפשר** Tech Support **לכל אדם ממישהו אחר בכל מקום בעולם.**

**בחרתי בפרויקט זה מכיוון שמאוד התחברתי לרעיון אני חושב שהוא יאתגר אותי וילמד אותי להתעסק בדברים שונים בתחום הסייבר שעדיין לא נגעתי בהם. בנוסף אני רואה שימוש אמיתי לפרויקט שלי בחיי היום יום של כל בן אדם.**

**אתגרים שצפויים לי בפרויקט הם מימוש שליטה על מחשב מרחוק, לממש יכולת לראות מסך של מישהו אחר בזמן אמת ולתקשר עם רכיבים במחשב כמו מקלדת ועכבר בלי** delay**. בנוסף היכולת של התוכנה לבצע כמה** sessions **של תקשורת שעוברת דרך השרת בשביל לאפשר לכמה מחשבים לבצע בקרה בו זמנית ושכל** sessions **יעבדו באופן רציף בלי** delay**.**

1. הגדרת הלקוח

המערכת מיועדת לכל מי שצריך תמיכה מרחוק ומוכן לסמוך על מישהו אחר לעזור לו בנוסף המערכת יכולה להיות בשימוש רב בחברה בינלאומית עם הרבה עובדים בכל העולם. הלקוחות של המערכת יהיו כל אחד שמבקש תמיכה או מציע תמיכה.

1. הגדרת יעדים/מטרות

* להוות מרכז תמיכה שמחבר בין AssistanceSeeker לבין Helper בצורה בטוחה.
* ליצור תשתית שתאפשר ל AssistanceSeeker לתמוך מרחוק במחשב של Helper בצורה נוחה, יעילה ומהירה.

1. בעיות, תועלות וחסכונות

הבעיה שאותה אנו מנסים לפתור עם מערכת זו היא היכולת לקבל תמיכה למחשב מרחוק, כלומר בלי צורך לצאת מהבית בשביל לקבל תמיכה. אנחנו מנסים להשיג דרך אמינה, מהירה וישירה לייצר תמיכה מרחוק לאנשים מכל העולם בלי הגבלה.

תועלתמרכזית אחת שיש למערכת זו היא היכולת לכמה אנשים לבצע תמיכה בו זמנית, התוכנה עובדת כך שאנחנו לא מוגבלים בכמות אנשים ואפשר שכמה אנשים יעזרו למחשבים אחרים בו זמנית.

תועלת נוספת שיש לפרויקט זה הוא שהשליטה של המחשב תלויה ב AssistanceSeeker בעיקר, כלומר אין מצב בו Helper יכול להשיג שליטה על מחשב ולא לנתק אותו כך שהוא יכול לבצע דברים שהמבקש תמיכה לא בסדר איתם, למבקש תמיכה תמיד יש את האופציה לנתק את הקשר. בנוסף כל קשר תלוי בזמן כלומר אם נראה לנו כי קשר מת (כלומר עבר כמות זמן מסוימת עד שה Helper לא ביצע משהו, הקשר מתנתק).

שירותים שהמערכת תיתן הם:

* מרכז תמיכה שמאפשר ל- Helper ול- AssistanceSeeker להתחבר אליו במטרה להבטיח מהימנות ה- Helper -ים.
* חיבור בין Helper ל- AssistanceSeeker ע"פ בקשת ה Helper כלומר ל Helper יש אפשרות לבחור מי תומך בו מבין ה- AssistanceSeeker הקיימים במערכת.
* תשתית ל- Helper שתאפשר לו לתמוך מרחוק ע"י שליטה במחשב ה – Helper.
* בקרה על ה"חיבור" כדי להבטיח את מהימנות התמיכה.

פתרונות ויישומים הקיימים:

TeamViewer- תוכנה זו מאפשרת לפתוח סאשנים של תמיכה מרחוק הנשלחים למבקש תמיכה או התומך על ידי קוד או אימייל. מול התוכנה שלי, ההבדל העיקרי הוא שהמערכת הזאת לא סגורה כלומר אפשר לעבוד איתה גם בלי להוריד את התוכנה למחשב.

<https://www.teamviewer.com/en-mea/>

AnyDesk- תוכנה זו מאפשרת תמיכה ושליטה בין שני מחשבים באופן מהיר ואמין כך שכל אחד יכול לקבל תמיכה מכל מקום בעולם.

<https://anydesk.com/en/solutions/remote-support>

1. האם צפויים קשיים או מגבלות בהגדרת המערכת

לא מדובר בטכנולוגיה חדשה. מאחר שהפרויקט מפותח ברשת הבית ספרית שהינה רשת סגורה כל משתמשי המערכת יהיו חייבים להיות משתמשים ברשת זאת לכן אין מגבלות הצפויות.

1. תיחום הפרויקט

הפרויקט עוסק בעיקר ברשתות ומערכות הפעלה.

רשתות – שרת מרובה לקוחות העברת כל האינפורמציה דרך הרשת.

מערכות הפעלה - מערכת הפעלה תצטרך לתת שליטה מלאה למחשב האחר לעבוד עליה ולתקשר עם הרכיבים הפנימיים של המחשב דרך הרכיבים החיצוניים של המחשב השני כלומר עכבר, מקלדת, מסך כמו כן שימוש ב thread-ים וב – Process-ים למקבול תהליכים.  
שימוש בהצפנה להבטחת קישורים ובבסיס נתונים לשמירת מידע של משתמשים.

המערכת לא מטפלת בתיקון בעיות פנימיות של המחשב, היא מטפלת רק ביצירת הקשר לבצע תמיכה מרחוק בין מחשבים.

1. קיימות

בעידן שלנו הרבה פעמים מחשבים נתקלים בבעיות כמו בעיות בתוכנה או ברכיבים הפיזיים של המחשב בדרך כלל שדברים כאלו קורים אנחנו קונים מחשב חדש או צריכים לקחת את המחשב לתיקון שמצריך הרבה כסף, זמן ומשאבים שפוגעים בכדור הארץ, כמו נסיעה במכונית. בנוסף, בהמון מצבים אפשר לפתור את הבעיה בלי אפילו לפתוח את המחשב ולחשוף את הרכיבים שלו. עם התוכנה שלי אפשר לפתור את כל הבעיות האלו בלי לבזבז כסף או לפגוע בכדור הארץ. כאשר אנו פועלים מהבית ולא שורפים דלק אנחנו עוזרים לכדור הארץ.

# פרק ב' - UELP - אפיון

1. פרוט המערכת:

**מערכת בקרת מחשב מרחוק היא תוכנה המאפשרת לאדם לסייע לאדם אחר ממחשב מרוחק.  
המערכת תחבר בין Helper (אנשים שמבינים במחשבים ומעוניינים לסייע לאחרים) לבין AssistanceSeeker (אנשים המעוניינים לקבל סיוע מאחרים)**

**המערכת תבטיח קבלת סיוע אך ורק מ- Helper-ים 'מאושרים' כדי שלא יגרם נזק ל- AssistanceSeeker.**

**המוצר הסופי אמור לאפשר Tech Support לכל אדם ממישהו אחר בכל מקום בעולם.**

**בחרתי בפרויקט זה מכיוון שמאוד התחברתי לרעיון אני חושב שהוא יאתגר אותי וילמד אותי להתעסק בדברים שונים בתחום הסייבר שעדיין לא נגעתי בהם. בנוסף אני רואה שימוש אמיתי לפרויקט שלי בחיי היום יום של כל בן אדם.**

**אתגרים שצפויים לי בפרויקט הם מימוש שליטה על מחשב מרחוק, לממש יכולת לראות מסך של מישהו אחר בזמן אמת ולתקשר עם רכיבים במחשב כמו מקלדת ועכבר בלי delay. בנוסף היכולת של התוכנה לבצע כמה** sessions **של תקשורת שעוברת דרך השרת בשביל לאפשר לכמה מחשבים לבצע בקרה בו זמנית ושכל** sessions **יעבדו באופן רציף בלי** delay**.**

1. מה היכולות שהיא תעניק למשתמש , פירוט היכולות

Helper-

* התחברות למערכת- מתחבר למערכת בעזרת שם משתמש סיסמא ואימייל.
* קבלת קוד זמני- מקבל קוד זמני מהמערכת להעביר למשתמש המבקש עזרה בשביל לפתוח את התמיכה ביניהם. (הקוד זמני)
* עצירת שיתוף- כאשר אנו רוצים לסגור את התמיכה לוחצים על הכפתור וחוזרים למצב של יכולת לפתוח סאשן תמיכה.
* שליטה על העכבר של ה AssistanceSeeker
* שליטה על המקלדת של ה AssistanceSeeker
* צפייה רציפה במסך ה- AssistanceSeeker

AssistanceSeeker-

* התחברות למערכת- מתחבר למערכת בעזרת שם משתמש סיסמא ואימייל.
* הכנסת קוד התחברות זמני- הכנסת הקוד של התומך בשביל להתחיל את התמיכה.
* עצירת שיתוף- כאשר אנו רוצים לסגור את התמיכה לוחצים על הכפתור וחוזרים למצב "מבקש תמיכה" שבו רואים את כל המשתמשים שמציעים תמיכה.

1. פירוט הבדיקות ('קופסא שחורה')

(בדיקות לביצוע אך ורק עם קובץ ההרצה, ללא הקוד, בדגש על כך שכל אחד לפי הכתוב בטבלה יצליח לבצע את הבדיקה)

| **מספר** | **שם הבדיקה (שם שיעיד על תוכן הבדיקה)** | **מה אמורה לבדוק** | **איך מתכננים לבדוק**  **(לתאר בפירוט את שלבי הבדיקה)** |
| --- | --- | --- | --- |
| 1 | פתיחת תמיכה בין שני משתמשים | יצירת תמיכה חדשה עם קוד מהתומך | התחברות למערכת על ידי שני מחשבים, אחד של מבקש תמיכה ואחד של תומך. התומך יוצר קוד והמבקש תמיכה מכניס את הקוד שנוצר לפני סיום הזמן המוגבל, בודקים שכל השליטה עברה לתומך והקוד באמת עבד |
| 2 | תמיכה עובדת עם מסך | בדיקה שרואים את המסך של המבקש תמיכה במסך של התומך | התחברות למערכת על ידי שני מחשבים, אחד של מבקש תמיכה ואחד של תומך. התומך יוצר קוד והמבקש תמיכה מכניס את הקוד שנוצר התומך בודק שכאשר הוא מתחבר הוא רואה את המסך של המבקש תמיכה כפי שהוא נראה באופן רציף גם כאשר יש שינויים |
| 3 | תמיכה עובדת עם עכבר | בדיקת תזוזת העכבר של התומך כעכבר של המבקש עזרה | התחברות למערכת על ידי שני מחשבים, אחד של מבקש תמיכה ואחד של תומך. התומך יוצר קוד והמבקש תמיכה מכניס את הקוד שנוצר. התומך בודק שכאשר הוא מזיז את העכבר שלו זה גם מזיז את העכבר במחשב של המבקש תמיכה ומגיב באופן רציף לתזוזות של העכבר |
| 4 | תמיכה עובדת עם מקלדת | בדיקת המקלדת של התומך כמקלדת של המבקש עזרה | התחברות למערכת על ידי שני מחשבים, אחד של מבקש תמיכה ואחד של תומך. התומך יוצר קוד והמבקש תמיכה מכניס את הקוד שנוצר. התומך בודק שכאשר הוא מקיש מקשים במקלדת שלו זה נקלט כהמקלדת של המבקש תמיכה ומבצע את הפעולה על מחשבו. |
| 5 | קיום תמיכה בין יותר מ2 מחשבים | בדיקה שאפשר לבצע תמיכה בין כמה מחשבים בו זמנית בלי הפרעה | התחברות למערכת על ידי 4 מחשבים, שניים של מבקשי תמיכה ושניים של תומכים. התומכים יוצרים קוד ומבקשי התמיכה מכניסים את הקודים הזמניים שנוצרו. אנו בודקים שכל הבדיקות 2-4 מתקיימות באופן רציף ומהיר על שני הסאשנים בלי בעיה או delay. |
| 6 | בדיקת פג תוקף של "קוד תמיכה" | בדיקה שהקוד תמיכה שיוצר התומך הוא זמני ולא עובד יותר מפרק זמן מסוים. | התחברות למערכת על ידי שני מחשבים, אחד תומך ואחד מבקש תמיכה, התומך יוצר קוד ומחכים עד שפרק הזמן שהקוד תקף בו ייגמר, רואים שהקוד לא עובד והוא כבר לא תקף בשביל המבקש תמיכה. |
| 7 | סיום תמיכה ע"י הנתמך | בדיקה שמבקש התמיכה מצליח לסיים במיידית תמיכה | התחברות למערכת על ידי שני מחשבים, אחד של מבקש תמיכה ואחד של תומך. התומך יוצר קוד והמבקש תמיכה מכניס את הקוד שנוצר. המבקש תמיכה בודק שכאשר הוא לוחץ על הכפתור לסיים את הבקרה, אז הבקרה באמת מסתיימת גם מהצד שלו וגם מהצד של התומך, כלומר כל השליטה חוזרת אליו. |
| 8 | סיום תמיכה ע"י התומך | בדיקה שנותן התמיכה מצליח לסיים במיידית תמיכה | התחברות למערכת על ידי שני מחשבים, אחד של מבקש תמיכה ואחד של תומך. התומך יוצר קוד והמבקש תמיכה מכניס את הקוד שנוצר. התומך בודק שכאשר הוא לוחץ על הכפתור לסיים את הבקרה, אז הבקרה באמת מסתיימת גם מהצד שלו וגם מהצד של המבקש תמיכה, כלומר כל השליטה חוזרת למבקש תמיכה. |
| 9 | ביצועים | בדיקה שרמת הביצועים לא מושפעת מכמות המשתמשים | התחברות למערכת על ידי 7+ מחשבים, יצירת כמה סאשנים תמיכה ובדיקה שלא נפגעת היכולות של המערכת עכב כמות האנשים. |
| 10 | יציבות | בדיקה שהמערכת לא קורסת בעת שימוש ביכולות שלה | בדיקת היכולות של המערכת כולל, התחברות, יצירת קוד, להיכנס לסאשן ולבצע שימוש ברכיבים החיצוניים של המחשב, הפסקת סאשן והתנתקות מהמערכת בעת כל נקודה בשימושה. |

1. תכנון לוח זמנים לפרויקט

| **פעילות** | **זמן התחלה מתוכנן** | **זמן סיום מתוכנן** | **זמן התחלה בפועל** | **זמן סיום בפועל** | **הערות** |
| --- | --- | --- | --- | --- | --- |
| יזום | 6.11.23 | 23.11.23 | 6.11.23 | 8.11.23 | סיום מוקדם |
| אפיון | 9.11.23 | 1.12.23 | 9.11.23 | 12.11.23 |  |
| ניתוח | 13.11.23 | 15.12.23 | 13.11.23 | 23.11.23 |  |
| עיצוב | 29.11.23 | 15.1.23 | 24.11.23 | 14.01.24 |  |
| קוד (לפרוט לאבנים גדולות) | בניית יכולת לשלוט על עכבר של מחשב אחד ממחשב אחר-  15.1.24  בניית יכולת לשלוט על מקלדת של מחשב אחד ממחשב אחר-  22.1.24  בניית יכולת לראות מסך של מחשב אחד ממחשב אחר-  30.1.24  העברת השליטה לבסיס שרת מרובה לקוחות (שרת מתקשר ומחבר בין מחשב אחד למחשב אחר לפי קוד מהנותן תמיכה)-  12.2.24  הוספת אימות נתונים (שם משתמש, סיסמא ואימייל) דרך database-  3.3.24  הצפנה עיצוב, תיקון באגים ושהמערכת עובדת תמיד-  28.3.24 | בניית יכולת לשלוט על עכבר של מחשב אחד ממחשב אחר-  22.1.24  בניית יכולת לשלוט על מקלדת של מחשב אחד ממחשב אחר-  30.1.24  בניית יכולת לראות מסך של מחשב אחד ממחשב אחר-  12.2.24  העברת השליטה לבסיס שרת מרובה לקוחות (שרת מתקשר ומחבר בין מחשב אחד למחשב אחר לפי קוד מהנותן תמיכה)-  3.3.24  הוספת אימות נתונים (שם משתמש, סיסמא ואימייל) דרך database-  28.3.24  הצפנה עיצוב, תיקון באגים ושהמערכת עובדת תמיד-  20.4.24 | בניית יכולת לשלוט על עכבר של מחשב אחד ממחשב אחר-  15.1.24  בניית יכולת לשלוט על מקלדת של מחשב אחד ממחשב אחר-  22.1.24  בניית יכולת לראות מסך של מחשב אחד ממחשב אחר-  05.02.24  העברת השליטה לבסיס שרת מרובה לקוחות (שרת מתקשר ומחבר בין מחשב אחד למחשב אחר לפי קוד מהנותן תמיכה)-  20.02.24  הוספת אימות נתונים (שם משתמש, סיסמא ואימייל) דרך database-  5.03.23  הצפנה עיצוב, תיקון באגים ושהמערכת עובדת תמיד-  26.03.24 | בניית יכולת לשלוט על עכבר של מחשב אחד ממחשב אחר-  22.1.24  בניית יכולת לשלוט על מקלדת של מחשב אחד ממחשב אחר-  04.02.24  בניית יכולת לראות מסך של מחשב אחד ממחשב אחר-  20.02.24  העברת השליטה לבסיס שרת מרובה לקוחות (שרת מתקשר ומחבר בין מחשב אחד למחשב אחר לפי קוד מהנותן תמיכה)-  3.3.24  הוספת אימות נתונים (שם משתמש, סיסמא ואימייל) דרך database-  25.03.24  הצפנה עיצוב, תיקון באגים ושהמערכת עובדת תמיד-  7.05.24 |  |
| גרסה ראשונית | 15.1.24 | 1.3.24 | 15.1.24 | 3.03.24 |  |
| מסמך בדיקות | 1.4.24 | 10.4.24 | 16.05.24 | 18.05.24 |  |
| מדריך למשתמש | 20.4.24 | 1.5.24 | 16.05.24 | 18.05.24 |  |
| גרסה סופית | 1.5.24 | 10.5.24 | 7.5.24 | 18.05.24 |  |
| סגירת תיק פרויקט | 1.5.24 | 10.5.24 | 7.5.24 | 18.05.24 |  |
| מצגת הסבר | 10.5.24 | 15.5.24 | 18.05.24 | 23.05.24 |  |

* **לוח הזמנים – הינו מסמך חי ויש לעדכנו לאורך כל הפרויקט**

1. ניהול סיכונים בפרויקט

* זיהוי סיכוני הפרויקט, ניתוחם ומענה עליהם, הטבלה תמוין מסיכון גבוה לנמוך (הגבוהים באדום, בינוני –כתום , קל – צהוב) סיכון שיוסר יצבע בירוק ויעבור לתחתית המסמך

| **הסיכון** | **פירוט הסיכון** | **רמת הסיכון**  **(קל/בינוני/קשה)** | **תיאור דרכים (לפחות 2 ) להתמודדות עם הסיכון ולהקטין אותו** | **מה בוצע בפועל** | **תאריך** |
| --- | --- | --- | --- | --- | --- |
| אי עמידה בזמנים | פרויקט לא יושלם | קשה | * הקדמת לו"זים משימות * ארגון הזמן בצורה יעילה * להתחיל בדברים היותר קשים | עמדתי בזמנים שהצבתי לעצמי והצלחתי לבצע ולסיים את הפרויקט בהצלחה | 16.05.24 |
| חוסר הצלחה לממש קליטה של המקלדת של המבקש תמיכה | פרויקט לא יושלם | קשה | * בדיקה באינטרנט * בדיקה לפני אם המימוש בכלל אפשרי * הסתייעות בחברים/מורה | יחסית התחלתי עם סוגייה זו אז טילפתי בה ומצאתי לה פתרון מהר כך שלא יצור בעיות בהמשך הפרויקט | 19/01/24 |
| חוסר הצלחה לממש קליטה של העכבר של המבקש תמיכה | פרויקט לא יושלם | קשה | * בדיקה באינטרנט * בדיקה לפני אם המימוש בכלל אפשרי * הסתייעות בחברים/מורה | יחסית התחלתי עם סוגייה זו אז טילפתי בה ומצאתי לה פתרון מהר כך שלא יצור בעיות בהמשך הפרויקט | 29/01/24 |
| חוסר הצלחה לממש שהתומך רואה את המסך של המבקש תמיכה | פרויקט לא יושלם | קשה | * בדיקה באינטרנט * בדיקה לפני אם המימוש בכלל אפשרי * הסתייעות בחברים/מורה | יחסית התחלתי עם סוגייה זו אז טילפתי בה ומצאתי לה פתרון מהר כך שלא יצור בעיות בהמשך הפרויקט | 19/02/24 |
| הבנת wxpython | עיכוב פרויקט | קל | * הבנה דרך גורמים מהאינטרנט * התייעצות עם חברים אחרים מהמגמה | כשהגעתי לגרפיקה חקרתי לעומק את המודול ולמדתי בעזרת חברים האינטרנט והמצגות | 29/03/24 |
| חוסר הצלחה לממש שרת מרובה לקחות | פרויקט לא שלם | קל | * התייעצות עם חברים מהמגמה * לא לעכב מימוש | מימשתי שרת מרובה לקוחות ובניתי את כל הפרויקט עליו כך שפתרתי את בעיה זו בהתחלה | 29/02/24 |
| בקרה איטית לתומך | פרויקט פחות שימושי | קשה | * חשיבה קדימה לממש את כל השלבים לפני בצורה הכי יעילה ככה הבקרה תהיה מהירה בסופו של דבר * התייעצות עם חברים מהמגמה | ביצעתי מספר הכי מינימלי של שליחות שעוברות ברשת כך שהתמיכה תהיה מהירה ויעילה | 03/04/24 |
| ביצועים | פרויקט פחות יעיל | קשה | * חשיבה קדימה לממש את כל השלבים בצורה הכי יעילה ונכונה מבחינת קוד, ככה שהביצועים של המערכת לא יפגעו * התייעצות עם חברים המגמה | ביצעתי מספר הכי מינימלי של שליחות שעוברות ברשת כך שהתמיכה תהיה מהירה ויעילה | 03/04/24 |

* **ניהול סיכונים – הינו מסמך חי ויש לעדכנו לאורך כל הפרויקט**

# פרק ג' - UELP – ניתוח

פרוט יכולות המערכת

**יכולות בצד שרת:**

שם היכולת: הרשמה למערכת

מהות היכולת: רישום משתמש חדש במערכת

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* פירק לפי פרוטוקול
* בדיקה מול בסיס הנתונים
* גיבוב סיסמא
* הוספה לבסיס נתונים
* בנה תשובה לפי פרוטוקול ובדיקה
* הצפנה
* שליחה

רשימת אובייקטים: הצפנה/פענוח, תקשורת, בסיס נתונים, פרוטוקול

שם היכולת: התחברות למערכת

מהות היכולת: התחברות עם משתמש קיים למערכת

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* פירק לפי פרוטוקול
* בדיקה מול בסיס הנתונים
* בנה תשובה לפי פרוטוקול ובדיקה
* הצפנה
* שליחה

רשימת אובייקטים: הצפנה/פענוח, תקשורת, בסיס נתונים, פרוטוקול

שם היכולת: בחירת סוג משתמש

מהות היכולת: בחירת סוג משתמש (עוזר או מבקש תמיכה)

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* פירק לפי פרוטוקול
* בדיקה מול דרישות כל סוג משתמש
* בנה תשובה לפי פרוטוקול ודרישות סוג המשתמש
* הצפנה
* שליחה

רשימת אובייקטים: הצפנה/פענוח, תקשורת, דרישות סוג משתמש, פרוטוקול

שם היכולת: בקשת פתיחת סאשן

מהות היכולת: יצירת קוד חדש לשני משתמשים במערכת להתחבר

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* פירק לפי פרוטוקול
* בדיקת סוג המשתמש ותקינות קבלת קוד
* בניית קוד משותף
* התחלת זמן תוקף הקוד ושמירת הקוד
* בנה תשובה לפי פרוטוקול ובדיקה עם קוד וזמן תוקף
* הצפנה
* שליחה

רשימת אובייקטים: הצפנה/פענוח, בניית קוד והגבלת זמנו, תקשורת, פרוטוקול

שם היכולת: בקשת התחברות לסאשן

מהות היכולת: התחברות לסאשן קיים בעזרת קוד זמני קיים

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* פירק לפי פרוטוקול
* בדיקת סוג המשתמש ותקינות הקוד
* בנה תשובה לפי פרוטוקול והבדיקה
* אם הקוד תקין - **חיבור שני משתמשים ביחד לקיום תמיכה**
* אם הקוד לא תקין, נשלחת הודעה אך ורק למזין הקוד:
  + בניית הודעת ע"פ פרוטוקול למבקש העזרה
* הצפנה
* שליחה

רשימת אובייקטים: הצפנה/פענוח, בדיקת קוד לפי קודים קיימים ותוקפם, תקשורת, פרוטוקול

שם היכולת: חיבור שני משתמשים ביחד

מהות היכולת: חיבור שני משתמשים ביחד לקיום תמיכה

אוסף יכולות:

* **בקשת פתיחת סאשן**
* **בקשת התחברות לסאשן**
* בניית הודעה לשני המשתמשים לפי פרוטוקולים עם מידע להתחברות אחד לשני
* הצפנה
* שליחה

רשימת אובייקטים: הצפנה/פענוח, תקשורת, פרוטוקול עם מידע לחיבור משתמש

**יכולות בצד לקוח:**

שם היכולת: הרשמה למערכת

מהות היכולת: רישום משתמש חדש במערכת (קליטת פרטיים אישיים נדרשים)

אוסף יכולות:

* ממשק משתמש – מסך הרשמה
* קליטת נתונים
* בדיקת תקינות
* בניית הודעה לפי פרוטוקול
* הצפנה
* שליחה לשרת
* קבלת תשובה מהשרת
* פענוח
* מפרק לפי פרוטוקול
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת, פרוטוקול

שם היכולת: התחברות למערכת

מהות היכולת: התחברות למערכת עם משתמש קיים (קליטת פרטיים אישיים נדרשים)

אוסף יכולות:

* ממשק משתמש – מסך התחברות
* קליטת נתונים
* בדיקת תקינות
* בניית הודעה לפי פרוטוקול
* הצפנה
* שליחה לשרת
* קבלת תשובה מהשרת
* פענוח
* מפרק לפי פרוטוקול
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת, פרוטוקול

שם היכולת: בחירת סוג משתמש (עוזר או מבקש תמיכה)

מהות היכולת: בחירת סוג משתמש (קליטת בחירה לפי כפתור)

אוסף יכולות:

* ממשק משתמש – מסך בחירה (כפתור למבקש תמיכה או תומך)
* קליטת לחיצת כפתור
* בניית הודעה לפי פרוטוקול
* הצפנה
* שליחה לשרת
* קבלת תשובה מהשרת
* פענוח
* מפרק לפי פרוטוקול
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת, פרוטוקול

שם היכולת: בקשת פתיחת סאשן עם קוד

מהות היכולת: בקשה מהשרת לקבל קוד זמני לפתיחת סאשן

אוסף יכולות:

* ממשק משתמש – מסך פתיחת סאשן
* קליטת לחיצה
* בניית הודעה לפי פרוטוקול
* הצפנה
* שליחה לשרת
* קבלת תשובה וקוד זמני מהשרת
* פענוח
* מפרק לפי פרוטוקול
* הצגת ממשק משתמש עם הקוד והזמן שיש למבקש תמיכה להתחבר

רשימת אובייקטים: ממשק משתמש- פתיחת סאשן, הצפנה/פיענוח, תקשורת, פרוטוקול, ממשק משתמש- הצגת קוד

שם היכולת: התחברות לסאשן עם קוד

מהות היכולת: בקשה מהשרת להתחבר לסאשן פתוח עם קוד מהתומך

אוסף יכולות:

* ממשק משתמש – מסך הכנסת קוד
* בדיקת קלט
* בניית הודעה לפי פרוטוקול
* הצפנה
* שליחה לשרת
* קבלת תשובה
* פענוח
* מפרק לפי פרוטוקול
* הצגת תשובה למשתמש

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת, פרוטוקול

שם היכולת: יצירת חיבור חדש מול עוד משתמש

מהות היכולת: חיבור למשתמש אחר במערכת בשביל התחלת תמיכה

אוסף יכולות:

* קבלת נתונים משרת
* פענוח
* מפרק לפי פרוטוקול
* יצירת חיבור חדש עם המשתמש

רשימת אובייקטים: הצפנה/פיענוח, תקשורת, פרוטוקול, יצירת חיבור חדש

יכולות בצד מבקש תמיכה:

שם היכולת: העברת תמונת מסך נוכחי

מהות היכולת: להעביר תמונה של מסך נוכחי לתומך

אוסף יכולות:

* לקחת תמונה של המסך הנוכחי
* להצפין את המידע של התמונה
* בניית מידע של התמונה לפי פרוטוקול
* שליחה לתומך את המידע

רשימת אובייקטים: הצפנה/פיענוח, תקשורת, פרוטוקול

שם היכולת: העברת מיקום עכבר חדש

שם היכולת: קבלת מיקום עכבר חדש מהתומך והזזת עכבר

מהות היכולת: לקבל את מיקום העכבר שהתומך הזיז אליו את העכבר ולהזיז בהתאם

אוסף יכולות:

* לקבל מידע מהתומך
* פענוח
* פירוק הפרוטוקול והוצאת מיקום עכבר חדש
* בדיקת מיקום
* הזזת עכבר למיקום שהתקבל

רשימת אובייקטים: הצפנה/פיענוח, תקשורת, פרוטוקול, בדיקת מיקום, הזזת עכבר למיקום שהתקבל

שם היכולת: קבלת תו נלחץ מהתומך ולחיצה עליו

מהות היכולת: לקבל תו שהתומך לחץ עליו

אוסף יכולות:

* לקבל מידע מהתומך
* פענוח
* פירוק הפרוטוקול והוצאת ייצוג התו
* בדיקת ייצוג התו
* לחיצה על התו אשר התקבל מהתומך

רשימת אובייקטים: הצפנה/פיענוח, תקשורת, פרוטוקול, בדיקת ייצוג התו, לחיצה על התו שהתקבל מהתומך

שם היכולת: ניתוק קשר מצד מבקש תמיכה

מהות היכולת: לנתק את הקשר במידה ומשתמש ביקש

אוסף יכולות:

* לקבל מהמשתמש לחיצה אשר אומר שהוא מנתק את הקשר
* בניית הודעה שאומרת למבקש תמיכה שמתנתק הקשר לפי פרוטוקול
* הצפנה
* שליחה למבקש תמיכה את ההודעה
* בניית ממשק משתמש בהתאם לניתוק

רשימת אובייקטים: הצפנה/פיענוח , ממשק משתמש , תקשורת, פרוטוקול, קליטת לחיצה לניתוק מהמשתמש

שם היכולת: ניתוק קשר מצד תומך

מהות היכולת: לנתק את הקשר במידה וקיבלנו כך מהתומך

אוסף יכולות:

* קבלת מידע מהתומך
* פענוח
* פירוק לפי פרוטוקול
* בניית ממשק משתמש בהתאם לניתוק

רשימת אובייקטים: הצפנה/פיענוח , ממשק משתמש , תקשורת, פרוטוקול

יכולות בצד תומך:

שם היכולת: קבלת תמונת מסך מהמבקש תמיכה

מהות היכולת: לקבל את תמונת המסך הנוכחית להצגה למשתמש

אוסף יכולות:

* לקבל את מידע של התמונה מהמבקש תמיכה
* פענוח
* פירוק הפרוטוקול
* בניית התמונה
* הצגת התמונה למשתמש

רשימת אובייקטים: הצפנה/פיענוח, תקשורת, פרוטוקול, הצגת תמונה

שם היכולת: להעביר מיקום עכבר חדש למבקש תמיכה במידה והשתנה

מהות היכולת: להעביר מיקום עכבר חדש במידה והוא השתנה

אוסף יכולות:

* לקבל מיקום של עכבר נוכחי במידה והשתנה
* בדיקה מיקום
* בניית הודעה עם מיקום עכבר לפי פרוטוקול
* הצפנה
* שליחה למבקש תמיכה את ההודעה

רשימת אובייקטים: הצפנה/פיענוח ,בדיקת מיקום, קבלת מיקום עכבר במידה והשתנה, תקשורת, פרוטוקול

שם היכולת: העברת לחיצת מקלדת/עכבר למבקש תמיכה

מהות היכולת: להעביר לחיצת מקלדת/עכבר למבקש תמיכה במידה ונלחצה

אוסף יכולות:

* לקבל תו שנלחץ על מקלדת/עכבר במידה ונלחץ
* בניית הודעה עם ייצוג התו לפי פרוטוקול
* הצפנה
* שליחה למבקש תמיכה את ההודעה

רשימת אובייקטים: הצפנה/פיענוח , קבלת תו במקלדת\עכבר במידה ונלחץ , תקשורת, פרוטוקול

# פרק ד' - UELP - העיצוב

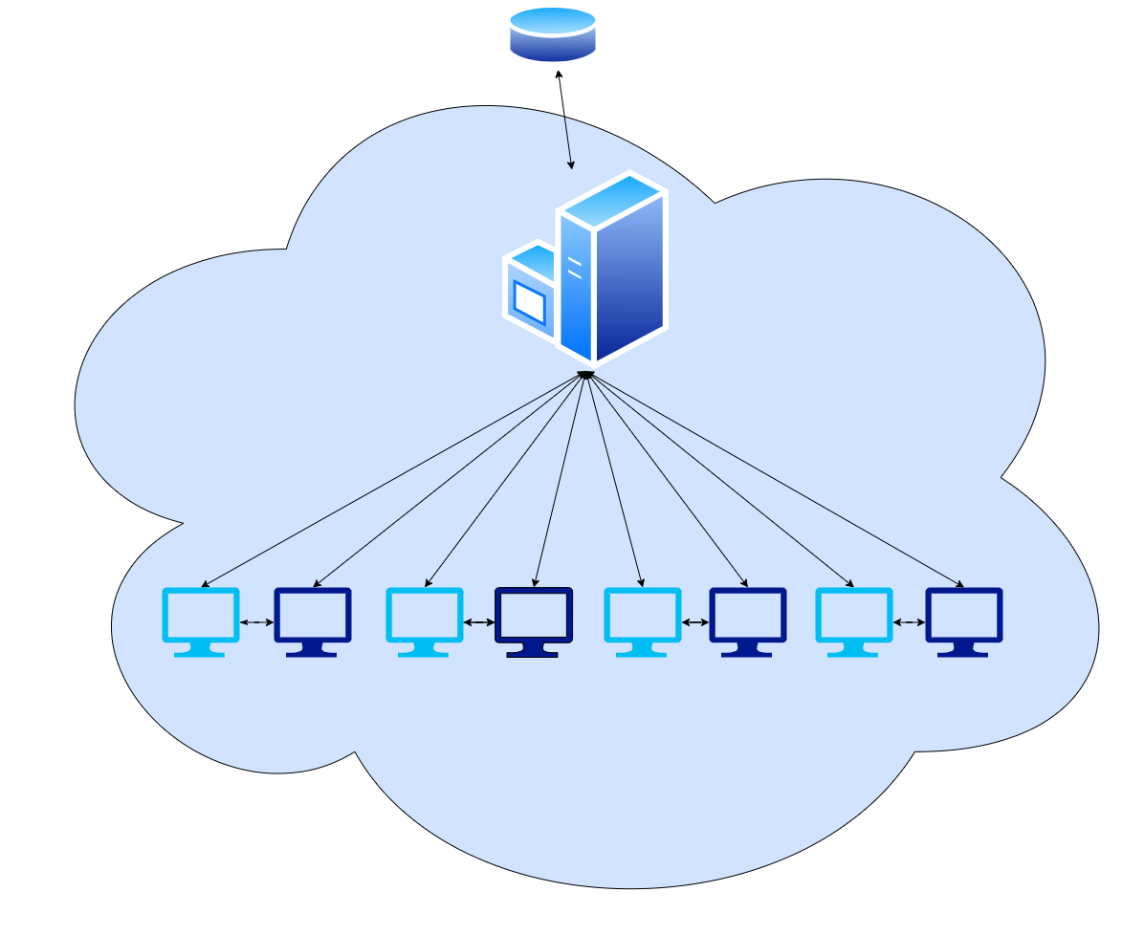
## תיאור הארכיטקטורה של המערכת המוצעת

תיאור החומרה:

* מחשבים עם מערכת הפעלה windows
* רשת אינטרנט
* בסיס נתונים מבוסס SQL

רכיבים שונים והקשרים ביניהם:

* שרת מרובה לקוחות (מחבר בין תומכים ונתמכים במערכת בעזרת קוד זמני)
* לקוחות (**נתמכים ותומכים**, המתחברים למערכת ומתחברים אחד לשני בעזרת השרת ואז מעבירים מידע ביניהם.)
* כל הלקוחות מעבירים מידע דרך רשת האינטרנט:
* שרת - לקוח
* שרת – בסיס נתונים
* תומך – נתמך



## תיאור הטכנולוגיה הרלוונטית

שפות תכנות: Python 3.7 ו-SQL לעבודה מול בסיס הנתונים

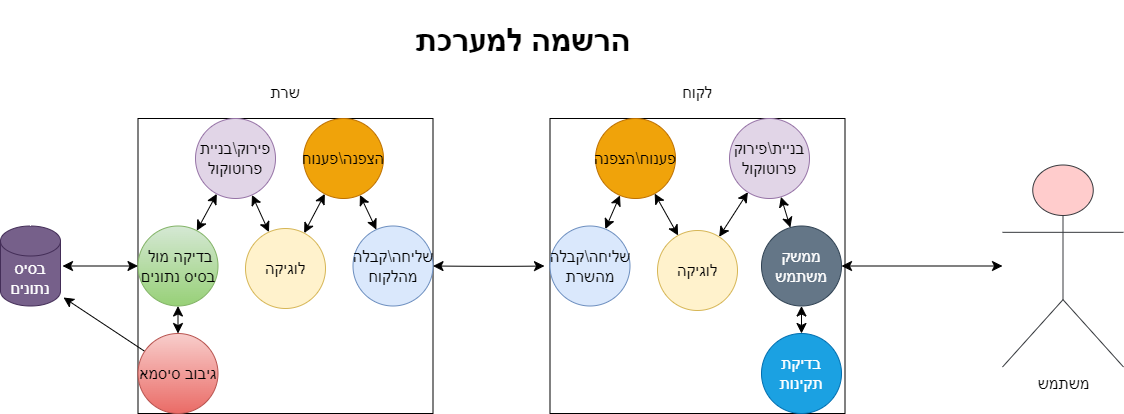
מערכת הפעלה: windows

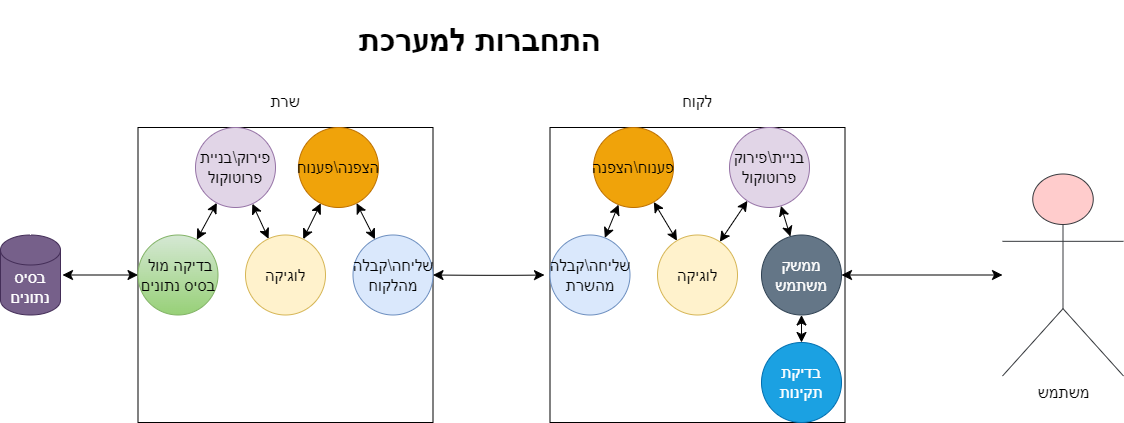
תקשורת: השרת הוא שרת מרובה לקוחות שמשתמש ב- sockets לתקשר עם הלקוחות, בנוסף ממומשת תקשורת peer2peer בין תומך לנתמך מבוססת סוקטים.

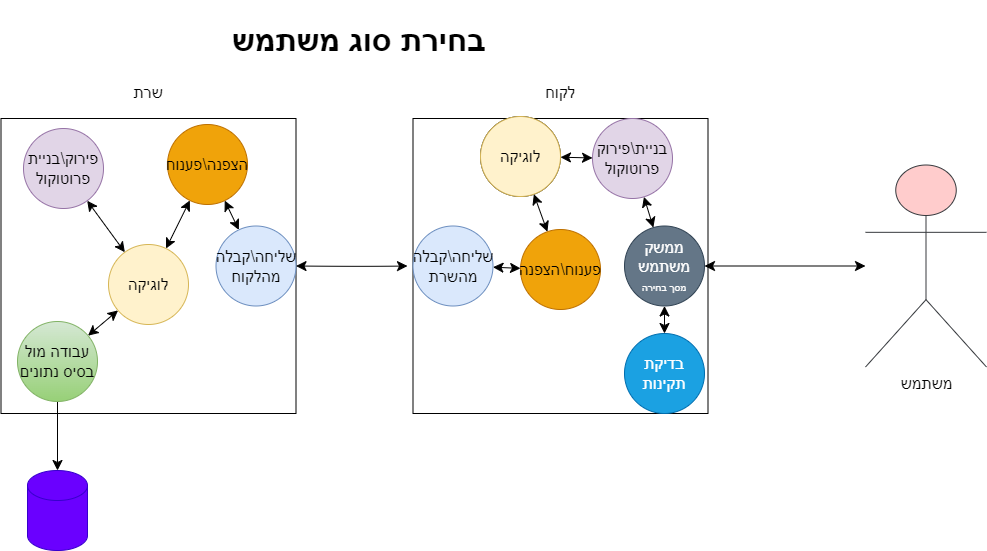
המידע יועבר ברשת בעזרת פרוטוקול TCP בשביל שהמידע יגיע מסודר, בנוסף המידע יהיה מוצפן לגמרי.

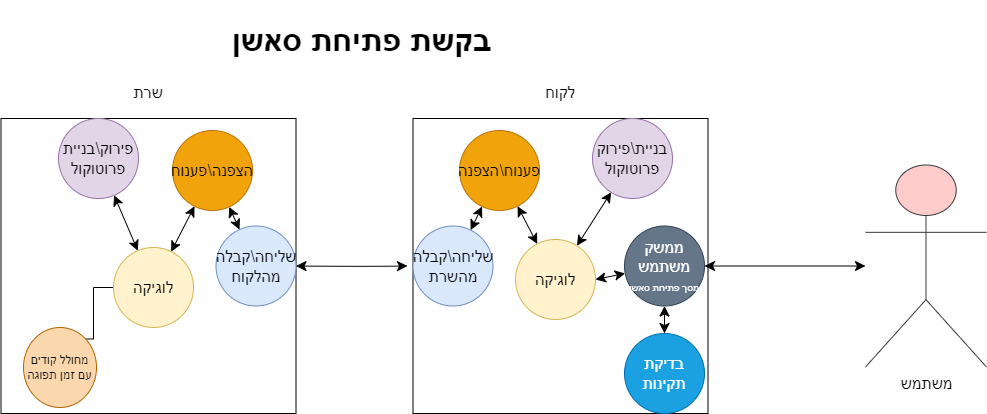
תחומי עניין: הצפנה, גרפיקה (באמצעות WXPython), עבודה עם בסיס נתונים (באמצעות SQL), שימוש ב thread-ים למיקבול פעולות במערכת, שימוש ב Process-ים למקבול תהליכים במערכת.

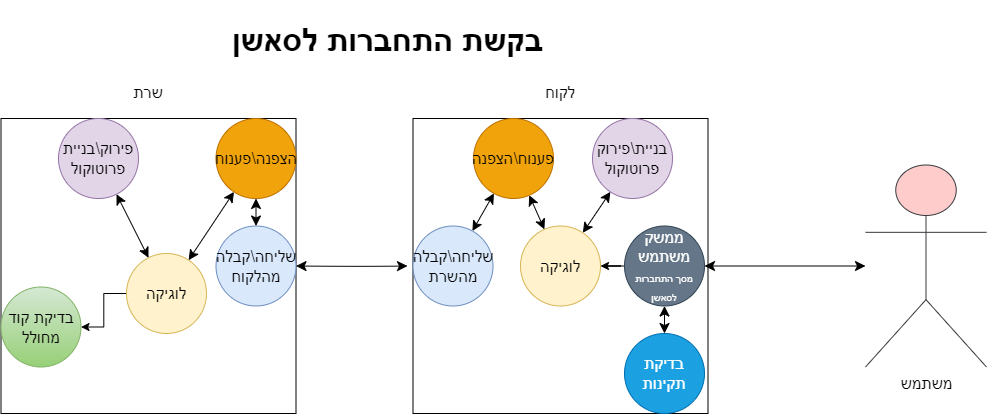
## תיאור מודולים בהם נעשה שימוש

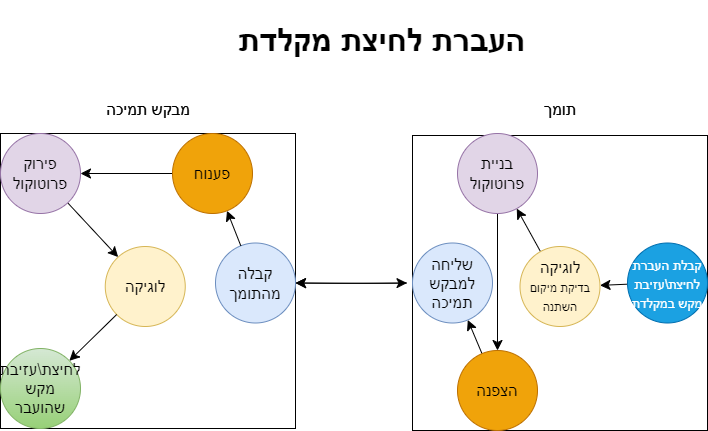
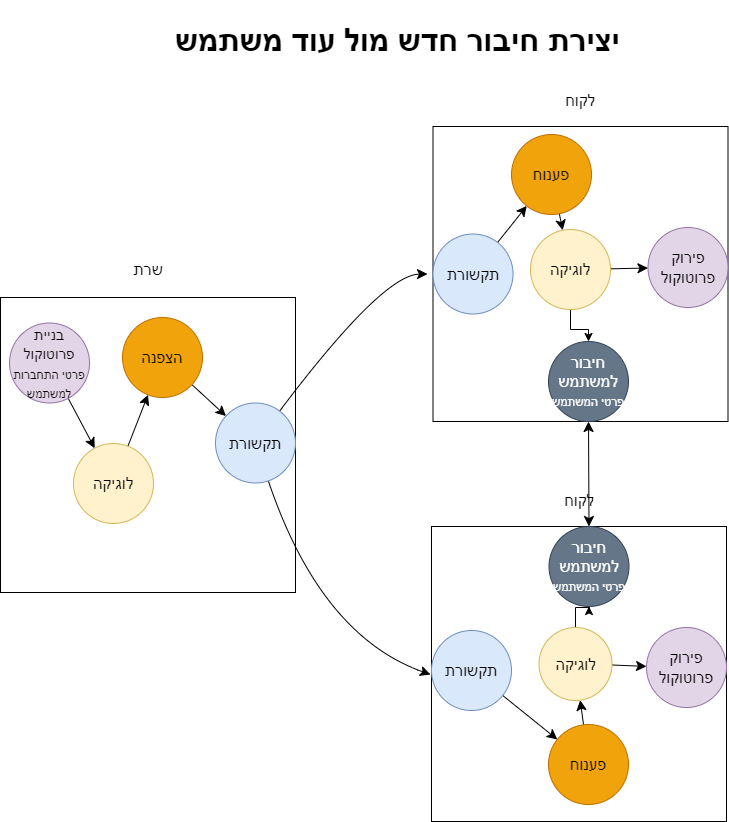


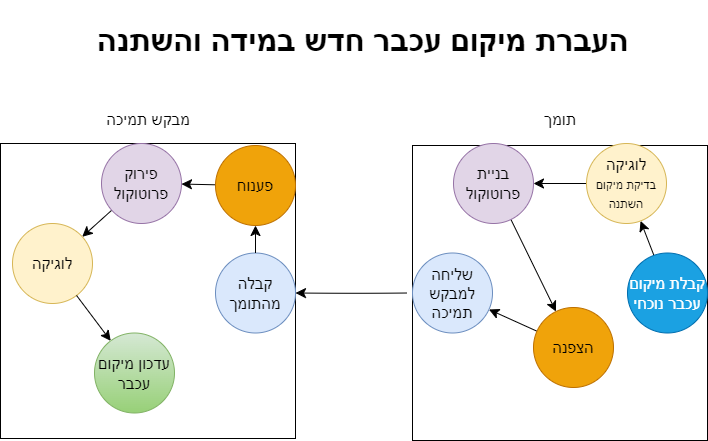


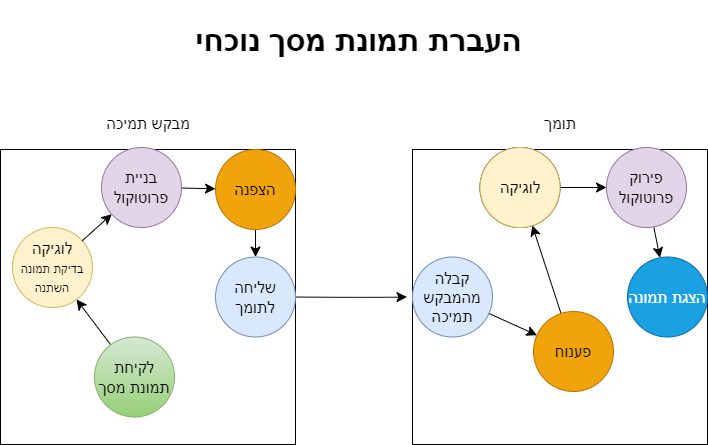
****

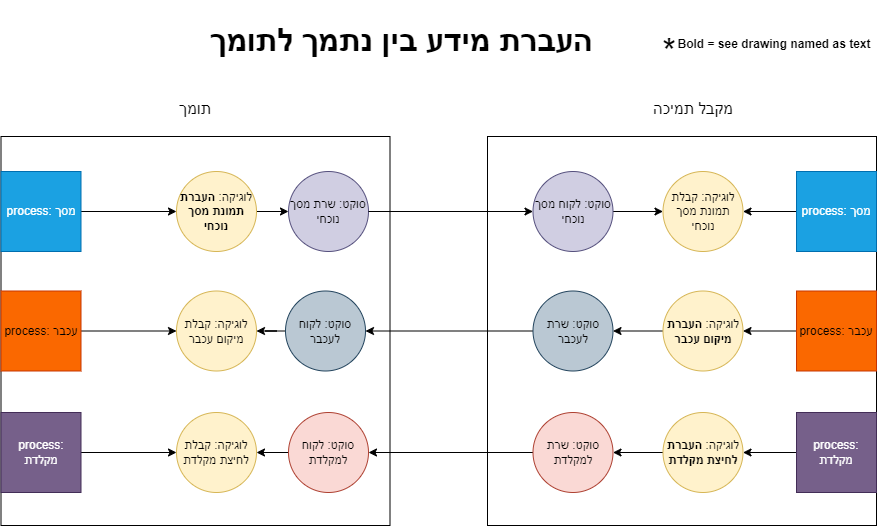
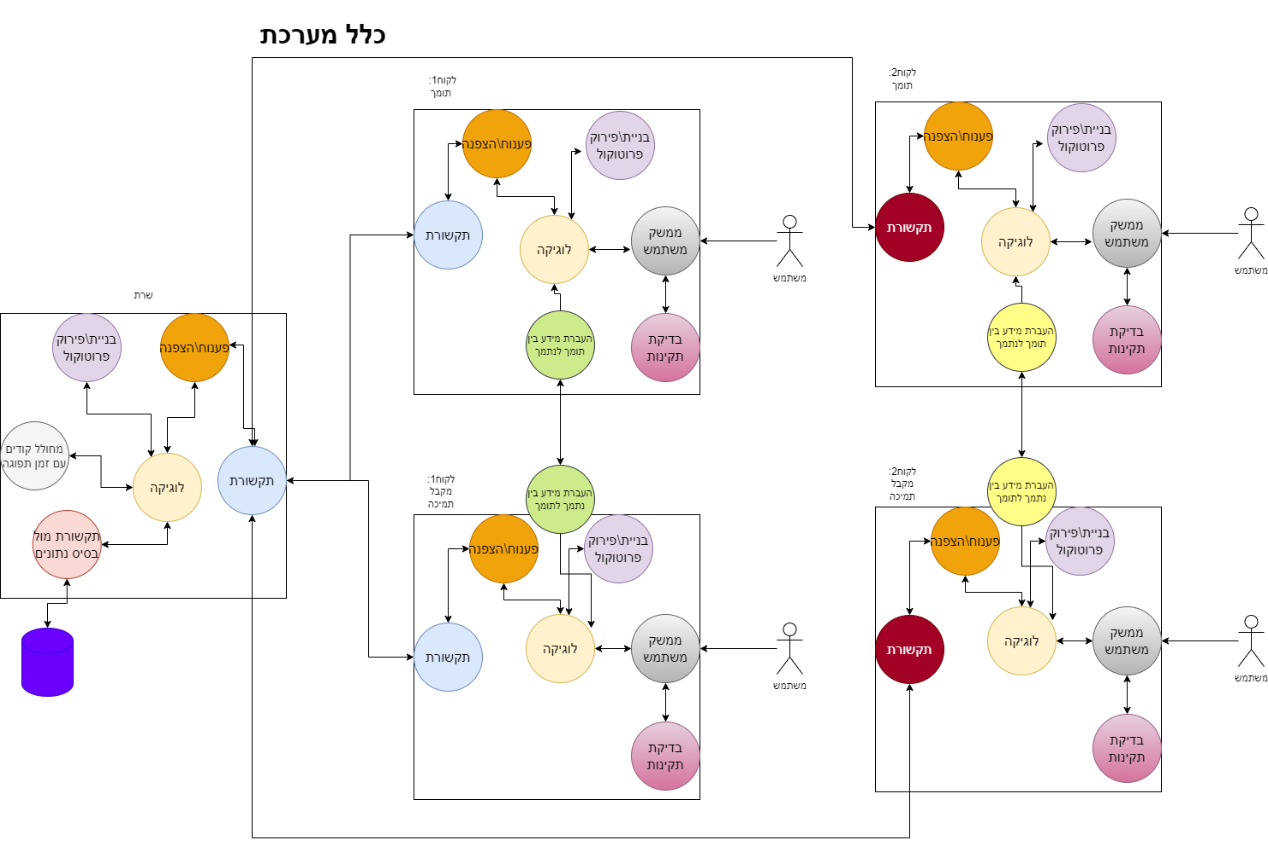
****

****

****



****

****

3.1 מודולים שאני מייבא:

| **שם המודול** | **לטובת** |
| --- | --- |
| queue | למימוש תור כמבנה נתונים |
| threading | לעבודה עם יותר מthread אחד במקביל |
| pynput | עבודה עם מעקב ושליטה על מקלדת ועכבר של מחשב (לחיצה על מקשים, הזזת ולחיצת עכבר ובדיקת מקשים ומיקומים של עכבר ומקלדת) |
| zlib | ביצוע פעולות compress וdecompress על מידע מועבר |
| PIL(ImageChops, ImageGrab, Image) | עבודה ועריכה עם תמונות, קבלת תמונות מהמחשב והדבקה על תמונות |
| pygame | להצגת מסך של הנתמך לתומך |
| cryptography | למימוש הצפנת rsa |
| multiprocessing | לעבודה והפעלת כמה פרוססים מפרוסס רץ |
| time | לעצירת פרוסס רץ לזמן מסוים |
| getnode (from UUid) | להשגת כתובת mac של המחשב |
| wx | יצירת גרפיקה של הפרויקט ועבודה עם גרפיקה ויצירת מסכים |
| pub (from pubsub) | עבודה לתקשורת בין threads |
| base64 | לencoding לbase64 ומימוש הצפנות |
| secrets | ליצירת רצף סטרינג רנדומלי למפתח וקוד סאשן |
| socket | עבודה של תקשורת בין שרת ללקוחות |
| sys | לעבודה עם פעולות של מערכת ההפעלה כמו יציאה מתוכנית |
| sqlite3 | לעבודה עם בסיס נתונים SQL |
| re | לחיפוש סוג מסוים של ערכים בתוך סטרינג למשל uppercase letters |
| pyperclip | להכניס ערך מסוים לclipboard של המחשב |
| wx.adv | לעבודה עם פעולות מתקדמות של גרפיקה |
| getmacbyip (from Scapy.layers.l2) | לשליחת בקשת arp לכתובת ip מסויימת |
| select | למימוש שרת מרובה לקוחות |
| hashlib | ליצירת ערך hash sha256 של סטרינגים. בשביל לשמור סיסמאות וכתובות mac בבסיס נתונים והשוואה עליהם |
| Random, AES, pad, unpad (from Cryptodome) | ליצירת הצפנת AES. מפתח הצפנה, הצפנה ופיענוח. |

3.2 מודולים שאני מייצר:

|  |
| --- |
| Server\_protocol |
| אוסף הפעולות הנדרשות בצד שרת לבנייה ופירוק הודעות ע"פ הפרוטוקול |

**אין למחלקה זו תכונות**

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| unpackData | הפעולה מקבלת מידע  מהתקשורת | הפעולה מפרקת את המידע שהתקבל לפרמטרים וה- opcode שלו.  מחזירה את הקוד הפעולה ואת השדות השייכים לקוד הפעולה.  (opcode, [parameters]) |  |
| pack\_login\_ans | הפעולה מקבלת סטרינג המייצג את סטטוס הצלחת ההתחברות למערכת | הפעולה מחזירה הודעה לפי פרוטוקול.    000 – הצליח  001 – נכשל  002- משתמש כבר מחובר  (000) |  |
| pack\_signup\_ans | הפעולה מקבלת סטרינג המייצג את סטטוס הצלחת הרשמה למערכת | הפעולה מחזירה הודעה לפי פרוטוקול.  010 – הצליח  011 – נכשל  (010) |  |
| pack\_typeuser\_ans | הפעולה מקבלת משתנה בוליאני המייצג אם בחירת המשתמש אושר או לא אושר | הפעולה מחזירה הודעה לפי פרוטוקול.  020 – הצליח  021 – נכשל  (020) |  |
| pack\_getcode\_ans | הפעולה מקבלת סטרינג המייצג את הקוד שמבקש המשתמש | הפעולה מחזירה הודעה לפי פרוטוקול.    0378634 – הצליח + קוד  (0378634) |  |
| pack\_expired\_code | הפעולה לא מקבלת כלום | הפעולה מחזירה הודעה לפי הפרוטוקול הודעת קוד לא תקף.  032- קוד לא תקף יותר |  |
| pack\_code\_ans | הפעולה מקבלת משתנה בוליאני המייצג את סטטוס אישור הקוד שהגיע מהלקוח | הפעולה מחזירה הודעה לפי פרוטוקול.    040 – עבר  041 – נכשל  (040) |  |
| pack\_con\_data | הפעולה מקבלת את הIP של משתמש והסוג משתמש  IP-  '127.0.0.1'  userType- “A” | הפעולה מחזירה הודעה לפי פרוטוקול.  A05127.0.0.1 – מידע על משתמש אחר  (A05127.0.0.1) |  |

|  |
| --- |
| Client\_protocol |
| אוסף הפעולות הנדרשות בצד שרת לבנייה ופירוק הודעות ע"פ הפרוטוקול |

**אין למחלקה זו תכונות**

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| unpackData | הפעולה מקבלת מידע מהתקשורת | הפעולה מפרקת את המידע שהתקבל לפרמטרים וה- opcode שלו.  מחזירה את הקוד הפעולה ואת השדות השייכים לקוד הפעולה.  (opcode, [parameters]) |  |
| pack\_mac\_addr | הפעולה מקבלת את הכתובתmac של המשתמש  macAddr- 00-B0-D0-63-C2-26 | הפעולה מחזירה הודעה לפי הפרוטוקול.  Opcode + mac address-  00-B0-D0-63-C2-2606 |  |
| pack\_login\_info | הפעולה מקבלת את פרטי ההתחברות של המשתמש: שם משתמש וסיסמא  loginInfo-  שם משתמש: itamar  סיסמא:  Itamar6 | הפעולה מחזירה הודעה לפי הפרוטוקול.  Opcode + login info-  006ItamarItamar6 |  |
| pack\_signup\_info | הפעולה מקבלת את פרטי ההרשמה של המשתמש.  signupinfo-ItamarItamar6 | הפעולה מחזירה הודעה לפי הפרוטוקול.  Opcode + signup info-  016ItamarItamar6 |  |
| pack\_type\_user | הפעולה מקבלת את סוג המשתמש שנבחר  typeUser-A/H | הפעולה מחזירה הודעה לפי הפרוטוקול.  Opcode + user type-  020 |  |
| pack\_code | הפעולה מקבלת קוד שהמשתמש נתמך הכניס  Code-a63h2j23 | הפעולה מחזירה הודעה לפי הפרוטוקול.  Opcode + code-  a63h2j2304 |  |

|  |
| --- |
| Server\_ comm |
| מחלקה ליצירת עצם לעבודה עם שרת מרובה לקוחות |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| recv\_q | Queue שמכניס לתוכו את כל ההודעות המתקבלות מהלקוחות ללוגיקה של השרת. |
| bindIP | הכתובת שממנה מקשיב השרת |
| Port | ה port שבו השרת רץ במחשב |
| socket | הsocket שממנו רץ השרת  (לא מתקבל מהמשתמש) |
| open\_clients | מילון של כל הsocket, ip שמחוברים לשרת וencryption object  Open\_clients[socket]=[ip, AESEncryption\_object]  (לא מתקבל מהמשתמש) |
| is\_running | משתנה בוליאני שאומר אם השרת כרגע רץ או לא  (לא מתקבל מהמשתמש) |
| RSAobject | עצם למימוש הצפנה א-סימטרית |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה מקבלת מידע ליצור עצם מסוג Server\_comm  Receive:  recv\_q: תור של הודעות  שדרכו יעבור ללוגיקה של השרת.  port: מספר פורט שעליו ירוץ השרת,  bindIP: כתובת ה-ip שהשרת מקשיב לו | הפעולה מאתחלת את תכונות המחלקה ומפעילה את \_main\_loop כ-thread שבו ירוץ השרת מרובה לקוחות |  |
| \_main\_loop | הפעולה לא מקבלת כלום | הפעולה יוצרת את השרת ומחכה שלקוחות יתחברו, (במידה והפורט הוא לא פורט ראשי מוודאים שהלקוח הוא p2pIP )כאשר לקוח חדש מתחבר לשרת מחליפים איתו מפתח באמצעות הפעולה \_get\_shared\_key (כthread)  ולאחר מכן מכל מידע שהלקוח מעביר לשרת היא מפענחת ומכניסה לrecv\_q עם הip של הלקוח במידה ומדובר בport המסך היא קוראת ל \_recvImage |  |
| \_get\_shared\_key | הפעולה מקבלת את ה ip של הלקוח החדש  Receive:  clientIP- הכתובת ip של הלקוח החדש שצריך לבנות איתו מפתח הצפנה משותף  curSocket-הסוקט של הלקוח שיש ליצור איתו מפתח | הפעולה שולחת ללקוח את המפתח הציבורי של ההצפנת RSA של השרת ובעזרת המפתח הלקוח שולח לשרת את המפתח המשותף שלהם מוצפן, הפעולה מכניסה לתוך מילון keys עצם הצפנה של AES של המפתח המשותף של הלקוח והשרת עם הip של הלקוח והip של הלקוח כהמפתח של המילון |  |
| \_disconnect\_client | הפעולה מקבלת socket של לקוח שמחובר לשרת ומנתקת אותו  Receive:  clientSocket- הסוקט של הלקוח שיש לנתק מהמערכת | הפעולה מוציאה את הלקוח משני המילונים שלה (open\_clients ו-keys)  וסוגרת ללקוח את הsocket  אם הלקוח הוא לקוח מסך מקלדת או עכבר סוגרים גם את השרת מכיוון שיש רק לקוח אחד. |  |
| \_find\_socket\_by\_ip | הפעולה מקבלת ip של לקוח במערכת ומחזירה את הsocket שלו  Receive:  findip- כתובת של לקוח במערכת | הפעולה מחזירה את הסוקט של הלקוח שהוכנס הip שלו  Return:  findSocket- הסוקט שחופש על ידי הכתובת שהתקבלה |  |
| send | הפעולה מקבלת ip של לקוח והודעה ושולחת את ההודעה המוצפנת ללקוח  Receive:  clientIP- כתובת של לקוח במערכת  msg- הודעה לא מוצפנת | הפעולה מצפינה את ההודעה שהתקבלה לפי מפתח ההצפנה (AES) ושולחת את האורך של ההודעה וההודעה שניהם דרך הsocket של הלקוח הזה |  |
| isRunning | הפעולה לא מקבלת כלום | הפעולה מחזירה את משתנה is\_running לבדוק אם השרת פועל  Return:  is\_running- משתנה במחלקה שאומר אם השרת שלה רץ או לא  (True) |  |
| recvImage\_ | הפעולה מקבלת socket ו-data | הפעולה מפענחת את המידע  מפרקת לפי פרוטוקול:  left, top, right, bottom, imgLen == server\_protocol.unpack(data)  בלולאה קולטת את המידע ע"פ imgLen  ומכניסה לתור ההודעות באופן הבא:  (ip, [top, left, bottom, right, bytearray]) | רלוונטי אך ורק לסוקט של התמונה |
| Close\_server | הפעולה לא מקבלת כלום | הפעולה מפסיקה את השרת הנוכחי, והופכת את משתנה is\_running לFalse |  |
| closeClient | הפעולה מקבלת כתובת ip של לקוח שהשרת רוצה לסגור  Receive:  clientIP- כתובת של לקוח במערכת | הפעולה מנתקת את הסוקט של המשתמש עם הip שהתקבל ומוציאה אותו מרשימות השרת בעזרת \_handle\_disconnect\_client |  |

|  |
| --- |
| Client\_ comm |
| מחלקה לעבודה ותקשורת עם שרת המערכת |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| serverIP | הכתובת ip של השרת |
| port | הport שעליו עובד השרת |
| recv\_q | תור של הודעות המתקבלות מהשרת |
| socket | הsocket של הלקוח |
| is\_running | משתנה בוליאני שאומר אם הלקוח רץ או לא |
| sharedKey | מפתח הצפנה משותף (AES) של הלקוח והשרת |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה מקבלת מידע ליצור עצם מסוג Client\_comm  Receive:  serverIP- כתובת הip של השרת  port- הפורט שעליו עובד השרת  recv\_q- תור של הודעות שדרכו יעבור ללוגיקה של השרת.  port: מספר פורט שעליו ירוץ השרת | הפעולה מפעילה את \_main\_loop כ-thread שבו ירוץ הלקוח |  |
| \_main\_loop | הפעולה לא מקבלת כלום | הפעולה מתחברת לשרת, ובונה מפתח הצפנה משותף לשרת וללקוח בעזרת \_get\_shared\_key  ולאחר מכן כל מידע שהשרת מעביר ללקוח היא מפענחת ומכניסה לrecv\_q |  |
| \_get\_shared\_key | הפעולה לא מקבלת כלום | הפעולה מקבלת מהשרת את המפתח הפומבי שלו ומשתמשת בו להצפין מפתח רנדומלי הלקוח משתמש במפתח זה ליצור עצם AES להצפנה ופיענוח |  |
| send | הפעולה מקבלת הודעה שהלקוח רוצה לשלוח לשרת  Receive:  msg– הודעה שהלקוח שולח לשרת ולא מוצפנת | הפעולה מצפינה את ההודעה בעזרת המפתח המשותף ושולחת את אורכה של ההודעה המוצפנת ואת ההודעה המוצפנת לשרת |  |
| sendImage | הפעולה מקבלת מידע על התמונה (data) ואת המידע של התמונה (imageData) | הפעולה מצפינה את הdata ושולחת לפי פרוטוקול ושולחת את הimageData מיד אחר כך (לא מוצפן) |  |
| close | הפעולה לא מקבלת כלום | הפעולה סוגרת ומנתקת את הsocket של הלקוח והופכת את משתנה is\_running לFalse |  |
| exchangeStatus | הפעולה לא מקבלת כלום | הפעולה מחזירה אמת אם נוצר כבר מפתח הצפנה משותף בין לקוח ושרת ושקר אם עדיין לא נוצר |  |

|  |
| --- |
| Database\_comm |
| מחלקה לעבודה ותקשורת עם בסיס נתונים |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| dbName | השם של בסיס הנתונים (Uelp\_db) שאנו רוצים להתחבר אליו |
| conn | חיבור לבסיס נתונים SQL |
| curr | מצביע לטבלה מסוימת |
| loginTableName | השם של הטבלה users שבה מאוחסן כל הפרטי כניסה של המשתמשים של המערכת |
| blacklistTableName | השם של הטבלה macs שבה מאוחסן כל כתובות ה mac של משתמשים שחסומים מלהיות תומכים |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה לא מקבלת כלום | הפעולה מאתחלת את תכונות המחלקה וקוראת לפעולה \_create\_db |  |
| \_create\_db | הפעולה לא מקבלת כלום | הפעולה מתחברת לבסיס נתונים ויוצרת את הטבלאות המבוקשות במידה והן כבר לא קיימות |  |
| checkUser­­\_ | הפעולה מקבלת  username- שם משתמש | הפעולה בודקת אם קיים משתמש כזה בטבלתusers  הפעולה מחזירה בהתאמה אמת או שקר אם המשתמש קיים במערכת או לא |  |
| addUser | הפעולה מקבלת  username- שם משתמש שרוצה להירשם למערכת  password- סיסמא של המשתמש שרוצה להרשם עם לחשבון חדש | במידה והשם משתמש לא קיים (\_checkUser) במערכת הפעולה מוסיפה אותו עם ה hash של הסיסמא שלו לתוך הטבלהusers ומחזירה אמת אם הצליחה להוסיף אותו אחרת שקר. |  |
| checkPassword | הפעולה מקבלת שם משתמש וסיסמא | במידה והשם משתמש קיים (\_checkUser) נבדקת ה hash של הסיסמא שהתקבלה מול ה hash של הבסיס נתונים.  הפעולה מחזירה אמת במידה והם שווים אחרת שקר |  |
| macExists | הפעולה מקבלת כתובת mac | הפעולה מחזירה אמת במידה והhash של הmac קיים בטבלה macs אחרת שקר |  |
| addBlackMac | הפעולה מקבלת כתובת mac | הפעולה מוסיפה את ה hash של ה mac לטבלה macs אם הוא לא קיים (\_macExists) |  |

|  |
| --- |
| AES\_hash\_cipher |
| מחלקה להצפנה ופענוח סימטרי ועבודה עם hash |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| bs | Block size שהמחלקה תצפין ותפענח בו |
| key | מפתח הצפנה שבעזרתו נצפין ונפענח מידע |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה מקבלת מידע ליצור עצם חדש של קריפטוגרפיה  Receive:  Key- מפתח להצפנה סימטרית | הפעולה יוצרת עצם חדש להצפנה פיענוח |  |
| encrypt | הפעולה מקבלת מידע בטיפוס בינארי ומצפינה אותו  Receive:  raw- מידע לא מוצפן | הפעולה מצפינה את המידע בעזרת הblock size ומפתח ההצפנה שהתקבל ומחזירה את המידע המוצפן בטיפוס בינארי  Return:  encrypted- מידע מוצפן |  |
| decrypt | הפעולה מקבלת מידע מוצפן מטיפוס בינארי ומפענחת אותו  Recieve:  enc- מידע מוצפן | הפעולה מפענחת את המידע בעזרת ה block size והמפתח ההצפנה בהתקבל ומחזירה את המידע המפוענח בטיפוס בינארי.  Return:  data- המידע המפוענח |  |
| hash | הפעולה מקבלת מידע  Recieve:  Data- מידע שרוצים להפעיל עליו פונקציית hash | הפעולה מפעילה על המידע פונקציית hash ומחזירה את ה hash של מידע זה בטיפוס בינארי.  Return-  hashedData- המידע שקיבלנו לאחר שהפעילו עליו פונקציית hash. | פעולה סטטית |

|  |
| --- |
| RSA\_cipher |
| מחלקה לעבודה עם הצפנה א-סימטרית |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| publicKey | מפתח הצפנה ציבורי (שמור בבינארי) |
| privateKey | מפתח פיענוח פרטי (שמור בבינארי) |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה לא מקבלת כלום | הפעולה מאתחלת את שדות המחלקה ויוצרת בהם מפתח ציבורי ופרטי של rsa ויוצרת עצם RSA חדש להצפנת מידע ופיענוח |  |
| encrypt | הפעולה מקבלת מידע ומפתח הצפנה ציבורי כString  Receive:  data- מידע להצפין  publicKey- מפתח הצפנה ציבורי שבעזרתו מצפינים מידע | הפעולה מצפינה את המידע שהתקבל בעזרת המפתח הציבורי ומחזירה את המידע המוצפן.  Return:  enc- מידע מוצפן | הפעולה היא פעולה סטטית |
| getStringKey | הפעולה אל מקבלת כלום | הפעולה מחזירה את מפתח ההצפנה הציבורי של העצם כstring  return:  stringKey- מפתח הצפנה ציבורי כstring |  |
| decrypt | הפעולה מקבלת מידע מוצפן מטיפוס בינארי  Receive:  data- מידע לפענח | הפעולה מפענחת את המידע שהוצפן בעזרת המפתח הציבורי ומחזירה את המידע המפוענח בטיפוס בינארי.  Return:  decData- המידע המפוענח |  |

|  |
| --- |
| Session\_codes |
| מחלקה לעבודה עם קודים של סאשנים |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| codes | מילון של כל הקודים שנפתחו, זמן הוספת הקוד והIP של מי שיצר אותם.  {'192.3.4.203', (123456, 17:56:59.23} |
| expired\_q | תור שמכניסים לתוכו ip של לקוח שקודו כבר לא תקף |
| cleanseTime | כל כמה זמן קוד פג תוקף |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה מקבלת את התור להודעות ליצור עצם חדש | הפעולה מאתחלת את כל תכונות המחלקה לפי הצורך ומריצה את \_remove\_expiredכthread |  |
| \_remove\_expired | הפעולה לא מקבלת כלום | הפעולה עוברת על מילון הקודים כל 2 שניות ובודקת אם יש קודים שזמנם לא תקף, במידה וכן מוציאה אותם מהמילון ומכניסה לexpired\_q את הip של הקוד |  |
| createCode | הפעולה מקבלת ip של לקוח | הפעולה יוצרת קוד רנדומלי באורך קבוע חדש מחזירה אותו ומוסיפה אותו למילון |  |
| checkCode | הפעולה מקבלת קוד באורך קבוע | הפעולה מחזירה אמת במידה והקוד קיים במילון ותקף ושקר אחרת |  |

|  |
| --- |
| Server\_logic |
| מחלקה לבניית הלוגיקה של השרת |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| server | שרת מרובה לקוחות לתקשורת |
| recv\_q | תור לקבלת הודעות מהלקוחות |
| expired\_q | תור לקבלת כתובות של משתמש שהקודים שלהם פג תוקפם |
| codes | עצם לעבודה עם קודים זמניים של לקוחות (גלובלי) |
| port | הפורט שעליו ירוץ השרת |
| commands | מילון לכל הפעולות הכלליות שבו לכל opcode יש שם של פעולה מתאימה |
| users | מילון שמצמיד בין כתובות ip כתובות mac של לקוחות הסוג של הלקוח והשם משתמש |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main | הפעולה לא מקבלת כלום | הפעולה יוצרת תור recv\_q וexpired\_q, יוצרת שרת server ועצם קודים clientCodes  מפעילה את handleMsgs כthread  מפעילה handleExpired כthread |  |
| handleMsgs | הפעולה מקבלת את עצם השרת והתור הודעות.  receive:  server- עצם השרת  recv\_q- תור ההודעות | הפעולה יוצרת מופע של DB\_comm (אי אפשר להעביר בסיס נתונים ל thread) ועוברת בלולאה אינסופית מוציאה הודעות מהתור recv\_q במבנה של tuple הכולל (ip,data)  מפרקת לפי פרוטוקול  (opcode, [params]) ומפעילה פעולה לפי הopcode :  commands[opcode](clientIP, DB, server, params) |  |
| handleExpired | הפעולה מקבלת את עצם השרת ותור הפג תוקף  server- עצם השרת  expired\_q- תור פג תוקף | הפעולה עוברת בלולאה אינסופית על תור הפג תוקף expired\_q ומוציאה כתובות ip  ושולחת הודעות לפי פרוטוקול ללקוחות  server.send(ip, server\_protocol.pack\_expired\_code()) |  |
| handle\_mac\_addr | הפעולה מקבלת את הip של הלקוח, העצם של הבסיס נתונים, עצם השרת ורשימה של פרמטרים שבתוכה יש את כתובת הmac של המשתמש.  clientIP- הכתובת של הלקוח  DB- עצם הבסיס נתונים  server- עצם השרת  params- רשימת פרמטרים | הפעולה מוציאה את כתובת הmac מהפרמטרים ומוסיפה את כתובת הip של הלקוח וכתובת הmac שלו למילון users  (users[clientIP] = (mac\_addr, None, None) |  |
| handle\_login | הפעולה מקבלת את הip של הלקוח, העצם של הבסיס נתונים, עצם השרת ורשימה של פרמטרים שבתוכה יש את שם משתמש והסיסמא  clientIP- הכתובת של הלקוח  DB- עצם הבסיס נתונים  server- עצם השרת  params- רשימת פרמטרים | הפעולה מוציאה את השם משתמש וסיסמא מהרשימה ובודקת אותם לפי הבסיס נתונים במידה והם תואמים מחזיר אמת אחרת שקר ושולחת הודעה בחזרה למשתמש לפי פרוטוקול ומוסיפה למילון משתמש את השם משתמש אם תואם  userOk = DB.checkPassword(username, password)  server.send(clientIP, pack\_login\_ans(userOk)) |  |
| handle\_signup | הפעולה מקבלת את הip של הלקוח, העצם של הבסיס נתונים, עצם השרת ורשימה של פרמטרים שבתוכה יש את שם משתמש והסיסמא  clientIP- הכתובת של הלקוח  DB- עצם הבסיס נתונים  server- עצם השרת  params- רשימת פרמטרים | הפעולה מוציאה את השם משתמש וסיסמא מהרשימה ובודקת אותם לפי הבסיס נתונים ומחזיר הודעה למשתמש ומוסיף למילון משתמשים בהתאם  newuserOk = DB.addUser(username, password)  server.send(clientIP, pack\_signup\_ans(newuserOk)) |  |
| check\_logged\_in | הפעולה מקבלת שם משתמש של משתמש | הפעולה מחזירה הפעולה בודקת אם השם משתמש נמצא בתוך מילון המשתמשים, ומחזירה אמת או שקר בהתאם. |  |
| handle\_typeUser | הפעולה מקבלת את הip של הלקוח, העצם של הבסיס נתונים, עצם השרת ורשימה של פרמטרים שבתוכה יש את סוג המשתמש (0/1(  clientIP- הכתובת של הלקוח  DB- עצם הבסיס נתונים  server- עצם השרת  params- רשימת פרמטרים | הפעולה מוציאה את בחירת סוג המשתמש מהפרמטרים בודקים אם הכתובת mac שהתקבל מהלקוח תואמת לכתובת הmac שמגיע מבקשת ה arp ששלחנו ללקוח. אם המשתמש רוצה להיות תומך, בודקים שכתובת הmac שלו לא בטבלת הblacklist בבסיס נתונים  במידה ולא נשמור את סוג המשתמש מוצמד עם הmac שלו,  שולחים הודעה מתאימה  If mac\_from\_arp == users[clientIP][0]:  If typeUser == “0”:  If not db.macExists(users[clientIP[0]):  Users[clientIP][1] = “H”  Typeuser\_ans = True  Else:  typeUser\_ans = False  elif typeUser == “1”:  users[clientIP][1] = “A”  typeUser\_ans = True |  |
| handle\_getCode | הפעולה מקבלת את הip של הלקוח, העצם של הבסיס נתונים, עצם השרת ורשימה של פרמטרים  clientIP- הכתובת של הלקוח  DB- עצם הבסיס נתונים  server- עצם השרת  params- רשימת פרמטרים | הפעולה בודקת את רשות המשתמש לקבלת קוד במידה ויכול שולחת לו קוד חדש לפי הפרוטוקול  if users[clientIP][1] == "H":  server.send(clientIP, pack\_getCode\_ans(True, codes.createCode(clientIP)) |  |
| handle\_codeCheck | הפעולה מקבלת את הip של הלקוח, העצם של הבסיס נתונים, עצם השרת ורשימה של פרמטרים עם הקוד  clientIP- הכתובת של הלקוח  DB- עצם הבסיס נתונים  server- עצם השרת  params- רשימת פרמטרים | הפעולה מוציאה את הקוד שהתקבל מרשימת הפרמטרים ובודקת אותו ושולחת הודעה חזרה למשתמש אם הקוד תואם. אם הקוד תואם תשלח לשני המשתמשים שהתחברו לסאשן את םרטי ההתחברות.  userCode = params[0]  if users[clientIP][1] == "A":  server.send(clientIP, pack\_code\_ans(codes.checkCode(userCode)) |  |
| disconnectClient | הפעולה מקבלת את הip של הלקוח שרוצים לנתק ועצם השרת | הפעולה מוציאה את המשתמש מרשימת users וקוראת ל closeClient לנתק אותו מהשרת |  |

|  |
| --- |
| Client\_logic |
| מחלקה לבניית הלוגיקה של הלקוח |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| client | לקוח לתקשורת עם שרת (client\_comm) |
| ip | כתובת שעליה רץ השרת |
| recv\_q | תור לקבלת הודעות מהשרת |
| port | הפורט שעליו רץ השרת |
| commands | מילון לכל הפעולות הכלליות שבו לכל opcode יש שם של פעולה מתאימה |
| allGraphics | עצם גרפיקה ראשי לעבודה עם גרפיקה |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main | הפעולה לא מקבלת כלום | הפעולה יוצרת תור recv\_q יוצרת לקוח client עם הip והport  מפעילה את handleMsgs כthread  ויוצרת עצם גרפיקה ומעפילה אותו בthread הראשי |  |
| handleMsgs | הפעולה מקבלת את עצם הלקוח והתור הודעות.  receive:  client- עצם הלקוח  recv\_q- תור ההודעות | הפעולה מוציאה מהתור בלולאה אינסופית על כל הtuple שהתקבלו.  מפרקת לפי פרוטוקול  (opcode, [params]) ומפעילה פעולה לפי הopcode :  commands[opcode](client, params) |  |
| send\_mac | הפעולה מקבלת את עצם הלקוח | הפעולה משיגה את הmac של המחשב ושולחת אותו לשרת |  |
| handle\_login\_ans | הפעולה מקבלת רשימה של פרמטרים  שבהם את אישור ההתחברות  params- רשימת פרמטרים | הפעולה מוציאה מהרשימה את אישור ההתחברות ומעבירה אותו לגרפיקה  wx.CallAfter(pub.sendMessage, "login\_ans", ans=login\_ans) |  |
| handle\_signup\_ans | רשימה של פרמטרים  שבהם את אישור ההרשמה  params- רשימת פרמטרים | הפעולה מוציאה מהרשימה את אישור ההרשמה ומעבירה אותו לגרפיקה  wx.CallAfter(pub.sendMessage, "signup\_ans", ans=signup\_ans) |  |
| handle\_typeUser\_ans | מקבלת רשימה של פרמטרים  שבהם את אישור בחירת המשתמש  params- רשימת פרמטרים | הפעולה מוציאה מהרשימה את אישור בחירת המשתמש ומעבירה אותו לגרפיקה  wx.CallAfter(pub.sendMessage, "typeUser\_ans", ans=typeUser\_ans) |  |
| handle\_getCode\_ans | מקבלת רשימה של פרמטרים  שבהם את הקוד  params- רשימת פרמטרים | הפעולה מוציאה מהרשימה את תשובת הבקשת קוד ומעבירה אותו לגרפיקה  wx.CallAfter(pub.sendMessage, "gotten\_code", ans=getCode\_ans) |  |
| handle\_code\_ans | מקבלת רשימה של פרמטרים בהם האם הקוד שנשלח תקין או לא  params- רשימת פרמטרים | הפעולה מוציאה מרשימת הפרמטרים את אישור הקוד שנשלח ומעבירה אותו לגרפיקה  wx.CallAfter(pub.sendMessage, "code\_ans", ans=code\_ans) |  |
| handle\_conData | מקבלת רשימה של פרמטרים בהם הכתובת של הלקוח השני  params- רשימת פרמטרים | הפעולה מוציאה מרשימת הפרמטרים את הip של הלקוח האחר והסוג משתמש של המשתמש הפעולה יוצרת תור הודעות להעברת הודעות בין הלוגיקה לפרוסס המקלדת, process לעכבר, מקלדת ומסך בהתאמה לפי סוג המשתמש. הפעולה מחכה 5 שניות לסיום מסך ההתחברות בגרפיקה ומפעילה את הפרוססים.  הפעולה גם תקרא לפעולה check\_closed לאחר הפעלת הפרוססים.  if my\_user\_Type == "H":  mouse = multiprocessing.Process(target=Helper\_logic.main\_Helper, args=(otherIP, 2001, None,))  keyboard = multiprocessing.Process(target=Helper\_logic.main\_Helper, args=(otherIP, 2002, close\_queue,))  screen = multiprocessing.Process(target=Helper\_screen\_logic.main\_Helper\_screen, args=(otherIP,)) wx.CallAfter(pub.sendMessage, "connecting\_session") |  |
| check\_closed | הפעולה מקבלת את תור ההודעות בין הלוגיקה ופרוסס המקלדת, פרוסס העכבר מקלדת ומסך.  (close\_queue, mouse, screen, keyboard) | הפעולה מוציאה מידע מתור ההודעות, אם מקבלת הודעה close (ההודעה היחידה שעוברת בתור) היא סוגרת את כל הפרוססים של השליטה ויוצאת מהפעולה |  |

|  |
| --- |
| Helper\_protocol |
| אוסף הפעולות הנדרשות בצד תומך לבנייה ופירוק הודעות ע"פ הפרוטוקול |

**אין למחלקה זו תכונות**

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| unpackData | הפעולה מקבלת מידע  מהתקשורת | הפעולה מפרקת את המידע שהתקבל לפרמטרים וה- opcode שלו לפי הפרוטוקול.  מחזירה את הקוד הפעולה ואת השדות השייכים לקוד הפעולה.  (opcode, [parameters]) | בשימוש רק להעברת תמונה |
| pack\_key\_click | הפעולה מקבלת unicode value של מקש שנלחץ | הפעולה מחזירה הודעה לפי הפרוטוקול  opcode + Unicode key-  01 unicode\_key.zfill(7) |  |
| pack\_key\_release | הפעולה מקבלת unicode value של מקש שנעזב | הפעולה מחזירה הודעה לפי הפרוטוקול  opcode + Unicode key-  02 unicode\_key.zfill(7) |  |

|  |
| --- |
| AssistanceSeeker\_protocol |
| אוסף הפעולות הנדרשות בצד תומך לבנייה ופירוק הודעות ע"פ הפרוטוקול |

**אין למחלקה זו תכונות**

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| unpackData | הפעולה מקבלת מידע  מהתקשורת | הפעולה מפרקת את המידע שהתקבל לפרמטרים וה- opcode שלו לפי הפרוטוקול.  מחזירה את הקוד הפעולה ואת השדות השייכים לקוד הפעולה.  (opcode, [parameters]) | בשימוש רק להעברת מקלדת |
| pack\_part\_image | הפעולה מקבלת את ערך המיקום של השמאל, למעלה, ימין ולמטה של התמונה אורך המידע של התמונה | הפעולה מחזירה הודעה לפי הפרוטוקול  opcode + top + left + bottom + right + image data size -  0103000400120010001234 |  |
| pack\_full\_image | הפעולה מקבלת את אורך המידע של התמונה | הפעולה מחזירה הודעה לפי הפרוטוקול  opcode + image data size -  02 img\_data\_size.zfill(10) |  |

|  |
| --- |
| Helper \_logic |
| מחלקה לבניית הלוגיקה של השרת של התומך |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| server | שרת מרובה לקוחות לתקשורת |
| recv\_q | תור לקבלת הודעות מהלקוחות |
| port | הפורט שעליו ירוץ השרת |
| otherIP | הip של הלקוח שאמור להתחבר לשרת |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main\_Helper | הפעולה מקבלת את כתובת הip של המבקש תמיכה הפורט שעלו עובדים והתור הודעות שעובר בין הלוגיקה למקלדת הודעות (None אם מופעל כפרוסס עכבר)  (otherIP, port, close\_queue) | הפעולה יוצרת תור ויוצרת בעזרתו שרת server שמקשיב רק לotherIP בפורט שהתקבל הפעולה יוצרת עצם מסוג Keyboard\_monitor/Mouse\_monitor ומאתחלת את תכונותיו בהתאמה (מעבירה לkeyboard\_monitor את close\_queue)  ומפעילה את פעולה check כthread בשביל לעבור על הודעות המתקבלות מלקוח המתחבר לשרתים של העכבר/מקלדת |  |
| check | הפעולה מקבלת את תור ההודעות ואת עצם ההקשה למקלדת או מסך שעובד על המחשב  (check\_queue, listener) | הפעולה מקבלת הודעות מהתור, אם התקבלה הודעת disconnect הלקוח התנתק מהשרת ומהסאשן לכן נעצור את עצם ההקשבה שעובד על המחשב ונצא מהפעולה |  |

|  |
| --- |
| Mouse\_monitor |
| מחלקה למעכב אחרי פעולות העכבר ועדכון עליהם |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| server | שרת מרובה לקוחות לתקשורת |
| ClientIP | כתובת לשלוח לה את מידע העכבר |
| listener | עצם של ההקשבה על העכבר |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה מקבלת עצם שרת וכתובת ip | הפעולה מאתחלת את תכונותיה וקוראת לפעולה \_monitor\_mouse |  |
| \_monitor\_mouse | הפעולה לא מקבלת כלום | הפעולה יוצרת listener חדש להקשיב לפעולות העכבר ומפעילה פעולות מתאימות בהתאמה |  |
| stop\_listening | הפעולה לא מקבלת כלום | הפעולה עוצרת את העצם הקשבה במידה והוא פועל |  |
| \_on\_click | הפעולה מקבלת את הx והy של הלחיצה ומידע על סוג ואופן הלחיצה | הפעולה נקראת כאשר מזוהה שנלחץ על העכבר הפעולה בודקת איזה סוג לחיצה זו ושולחת הודעה מתאימה ללקוח לפי פרוטוקול |  |
| \_on\_scroll | הפעולה מקבלת את הx והy של הגלגל והdx וdy של אופן שינוי הגלגל | הפעולה נקראת כאשר מזוהה שהגלגל של העכבר בפעולה, הפעולה בודקת איזה סוג תזוזה זו ושולחת הודעה מתאימה ללקוח לפי הפרוטוקול |  |
| \_on\_move | הפעולה מקבלת את הx והy של מיקום העכבר החדש | הפעולה נקראת כאשר מזוהה שהעכבר זז הפעולה בודקת איזה סוג תזוזה זו ושולחת הודעה מתאימה לפי הפרוטוקול |  |

|  |
| --- |
| AssistanceSeeker \_mouse\_logic |
| מחלקה לבניית הלוגיקה של הלקוח לעכבר |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| client | לקוח לתקשורת עם שרת (client\_comm) |
| otherIP | כתובת שעליה רץ השרת |
| recv\_q | תור לקבלת הודעות מהשרת |
| port | הפורט שעליו רץ השרת |
| mouse\_cont | עצם ששולט על העכבר |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main\_AS\_mouse | הפעולה מקבלת את כתובת הip של התומך | הפעולה יוצרת תור ויוצרת בעזרתו לקוח client עם otherIP וport  הפעולה עוברת על תור ההודעות בלולאה אינסופית ומקבלת מידע על העכבר ומבצעת את הפעולה של העכבר שנשלחה לפי הפרוטוקול |  |

|  |
| --- |
| Keyboard\_monitor |
| מחלקה למעכב אחרי פעולות המקלדת ועדכון עליהם |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| server | שרת מרובה לקוחות לתקשורת |
| ClientIP | כתובת לשלוח לה את מידע העכבר |
| special\_keys\_mapping | מילון שמצמיד בין מקשים מיוחדים לערכים חדשים בשבילהם |
| close\_queue | תור multiprocessing להעביר הודעות בין הלוגיקה לעצם הקשבה עכבר |
| listener | עצם ההקשבה למקלדת |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| \_\_init\_\_ | הפעולה מקבלת עצם שרת כתובת ip ותור ההודעות בין הלוגיקה למחלקה | הפעולה מאתחלת את תכונותיה וקוראת לפעולה  \_monitor\_keyboard |  |
| \_monitor\_keyboard | הפעולה לא מקבלת כלום | הפעולה יוצרת listener חדש להקשיב לפעולות המקלדת ומפעילה פעולות מתאימות בהתאמה |  |
| pressed\_end | הפעולה לא מקבלת כלום | הפעולה נקראת כאשר נלחץ על המקלדת של המחשב Shift + ctrl + d  קומבינציה זאת אומרת שהמשתמש רוצה להתנתק מהסאשן. הפעולה הפעולה שולחת לתומך הודעה לעזוב את המקשים האלו ועוצרת את עצם ההקשבה שלה. בנוסף הפעולה מכינסה close לתוך תור ההודעות close\_queue בשביל להודיע ללוגיקה לעצור את כל הפרוססים של השליטה. |  |
| stop\_listening | הפעולה לא מקבלת כלטם | הפעולה עוצרת את עצם ההקשבה למקלדת אם הוא פועל |  |
| \_on\_press | הפעולה מקבלת את הkey שנלחץ | הפעולה נקראת כאשר מזוהה שנלחץ מקש על המקלדת הפעולה ושולחת הודעה מתאימה ללקוח לפי פרוטוקול עם ייצוג המקש |  |
| \_on\_release | הפעולה מקבלת את הkey | הפעולה נקראת כאשר מזוהה ששוחרר מקש על המקלדת הפעולה ושולחת הודעה מתאימה ללקוח לפי פרוטוקול עם ייצוג המקש |  |

|  |
| --- |
| AssistanceSeeker \_keyboard\_logic |
| מחלקה לבניית הלוגיקה של הלקוח למקלדת |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| client | לקוח לתקשורת עם שרת (client\_comm) |
| otherIP | כתובת שעליה רץ השרת |
| recv\_q | תור לקבלת הודעות מהשרת |
| port | הפורט שעליו רץ השרת |
| Keyboard\_cont | עצם מסוג controller של pynput לעבודה עם מקלדת |
| close\_queue | תור multiprocessing להעביר הודעות בין הלוגיקה לפרוסס שמקבל את מידע המקלדת מהתומך |
| special\_keys\_mapping | מילון שמצמיד בין ערכים למקשים מיוחדים שלהם |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main\_AS\_keyboard | הפעולה מקבלת את כתובת הip של התומך ותור ההודעות להודעות בין הלוגיקה של הלקוח לפרוסס שמקבל את ההודעות מקלדת מהתומך | הפעולה יוצרת תור ויוצרת בעזרתו לקוח client עם otherIP וport  הפעולה מפעילה כ thread את פעולה check\_close לבדיקת לחיצה על קומבינצית ההתנתקות.  הפעולה עוברת על תור ההודעות בלולאה אינסופית ומקבלת מידע מפרקת אותו (AssistanceSeeker\_protocol) lומבצעת את הפעולה הנדרשת |  |
| check\_close | הפעולה מקבלת את תור ההודעות close\_queue ועצם הלקוח | הפעולה בודקת בלולאה אינסופית אם נלחץ קומבינצית הסגירת סאשן על המקלדת, אם נלחץ תסגור את הלקוח ותכניס לתוך תור ההודעות close בשביל להודיע ללוגיקה לסגור את כל הפרוססים של השליטה. |  |

|  |
| --- |
| AssistanceSeeker\_screen\_logic |
| מחלקה לבניית הלוגיקה של הלקוח של הנתמך להעברת המסך |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| client | עצם לקוח לתקשורת לשרת(client\_comm) |
| recv\_q | תור לקבלת הודעות מהשרת |
| port | הפורט שעליו רץ השרת |
| otherIP | הip של השרת |
| currScreen | עצם מסוג תמונה (PIL) של המסך כרגע |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main\_AS\_screen | הפעולה מקבלת את כתובת ה ip של התומך | הפעולה יוצרת תור ויוצרת בעזרתו לקוח client שמתחבר לשרת שרץ על otherIP בפורט שהתקבל ושולחת תמונת מסך ראשונית  בלולאה אינסופית הפעולה מבצעת תמונת מסך הפעולה קוראת לgetChanged במידה ויש שינוי שולחת את הריבוע המייצג את השינוי במידה והוא גדול מ75 אחוז שולחת את תמונת המסך החדשה  ושומרת אותה במשתנה התמונה |  |
| getChanged | הפעולה מקבלת תמונת מסך חדשה  ותמונת מסך הנוכחית | הפעולה מחזירה את מיקום ריבוע השינוי בתמונה ואת תמונת של השינוי עצמה |  |
| getChangedPrecentage | הפעולה מקבלת את הbounding box של תמונת השינוי, כלומר גודל תמונת השינוי ומספר הפיקסלים שיש על המסך | הפעולה מחשבת את אחוז השינוי מהמסך כולו לפי גודל תמונת השינוי וכמות הפיקסלים במסך |  |

|  |
| --- |
| Helper\_screen\_logic |
| מחלקה לבניית הלוגיקה של השרת למסך |

| תכונות | |
| --- | --- |
| שם תכונה | פירוט |
| server | לקוח לתקשורת עם לקוח(server\_comm) |
| otherIP | כתובת שעליה רץ השרת |
| recv\_q | תור לקבלת הודעות מהלקוח |
| port | הפורט שעליו רץ הלקוח |
| currScreen | עצם מסוג תמונה (PIL) של המסך כרגע (משתנה גלובלי) |

| פעולות | | | |
| --- | --- | --- | --- |
| שם פעולה | טענת כניסה | טענת יציאה | הערות |
| main\_Helper\_screen | הפעולה מקבלת את כתובת ה ip של התומך | הפעולה יוצרת תור ויוצרת בעזרתו שרת server עם otherIP וport ומשתנה למסך הנוכחי  הפעולה עוברת על תור ההודעות בלולאה אינסופית ומקבלת מידע מפרקת אותו בונה את התמונה וקוראת לbuildScreen במידה והתמונה היא לא תמונה מלאה אלא שינוי.  הפעולה מראה את התמונה בעזרת pygame ומראה אותה מחדש במידה וקיבלנו תמונה חדשה |  |
| build\_Screen | הפעולה מקבלת את הtop left right bottom ערכים של תמונת השינוי ואת תמונת השינוי עצמה בבתים | הפעולה בונה את המסך החדש בעזרת התמונה הקודמת של המסך והתמונת שינוי שהתקבלה ושומרת אותה בcurrScreen |  |

|  |
| --- |
| graphics |
| מחלקות ליצירת מסכי הגרפיקה למשתמש |

| שם מחלקה | שימוש המחלקה | פעולות חשובות במחלקה |
| --- | --- | --- |
| MyFrame | Frame ראשי לכל המסכים, עליו יוצרים את הPanel הראשי שבעזרתו מראים את כל ה panels של המסכים |  |
| MainPanel | הPanel הראשי, ניצור בתוך הפאנל הזה את כל הפאנלים של המסכים האחרים ונחליף ביניהם על הפאנל הזה | הפעולה change\_screen מקבלת את פאנל המסך הנוכחי ופאנל המסך להחליף אליו ומחליפה את הפאנל כך שתציג את המסך החדש |
| LoginPanel | המחלקה יוצרת את פאנל המסך התחברות.  במסך זה אנחנו מכניסים את השם משתמש וסיסמא להתחבר למערכת. כאשר לוחצים על כפתור ההתחברות השם משתמש וסיסמא נשלחים לשרת ואנו מקבלים בחזרה תשובה. אם התחברנו בהצלחה נעבור למסך הבחירת סוג משתמש ואם לא דבר זה יוצג למשתמש. יש אופציה בנוסף לעבור למסך ההרשמה למערכת בלחיצה על כפתור | הפעולה show\_invalid\_message מוסיפה טקסט בצבע אדום לSizer כך שיוצג על המסך למשך זמן מסוים ואז יעלם. נשתמש בפעולה זו להציג הודעות שגיאה למשתמש לגבי המידע שהכניס |
| SignUpPanel | המחלקה יוצרת את פאנל מסך ההרשמה.  במסך זה אנחנו מכניסים שם משתמש וסיסמא להירשם למערכת. כאשר לוחצים על כפתור ההרשמה המידע שהכנסנו נשלח לשרת אנו מקבלים תשובה. אם נרשמנו למערכת בהצלחה נעבור למסך הבחירת סוג משתמש אם לא הדבר יוצג למשתמש ויצטרך להכניס שם משתמש וסיסמא חדש בהתאם.  יש אופציה בנוסף לעבור למסך ההתחברות למערכת בלחיצה על כפתור | הפעולה show\_invalid\_message מוסיפה טקסט בצבע אדום לSizer כך שיוצג על המסך למשך זמן מסוים ואז יעלם. נשתמש בפעולה זו להציג הודעות שגיאה למשתמש לגבי המידע שהכניס |
| SelectUserPanel | המחלקה יוצרת את פאנל מסך הבחירת סוג משתמש.  במסך זה המשתמש יכול ללחוץ על 2 כפתורים או תומך או מבקש תמיכה.  לפי הכפתור שהמשתמש נלחץ השרת מחזיר תשובה האם המשתמש יכול להיות סוג המשתמש הזה. אם המשתמש אושר נעבור למסך הסוג משתמש הנבחר, אם לא נציג זאת למשתמש והוא יצטרך לבחור שוב. |  |
|  |  |  |
| HelperPanel | המחלקה יוצרת את פאנל המסך תומך. במחלקה זו ניצור מסל שבו נראה את הקוד סאשן שלנו. אנו מקבלים קוד מהשרת ברגע שאנו עוברים למסך זה. במידה והקוד נהיה פג תוקף נקבל הודעה לו כך מהשרת ויקפוץ פופ אפ למשתמש אם הוא רוצה קוד חדש או לא. אם לא, יחזור למסך בחירת סוג משתמש אם כן, יקבל קוד חדש ויראה אותו על המסך.  בנוסף המשתמש יכול ללחוץ על כפתור העתק שיש על המסך, בשביל להעתיק את הקוד לclipboard של המחשב שלו.  כאשר מבקש תמיכה מתחבר לסאשן עם הקוד נעבור אוטומטית למסך ההתחברות | הפעולה handle\_code\_gotten  נקראת כאשר מגיע הודעת קוד מהשרת, נבדוק אם ההודעה מתייחסת לקוד פג תוקף או קוד חדש. במידה וזה קוד חדש נציג אותו על המסך, במידה ולא נשאל את המשתמש אם הוא רוצה קוד חדש ונפעל בהתאם. |
| ASPanel | המחלקה יוצרת את פאנל מסך המבקש תמיכה. במחלקה זו ניצור מסך שבו נכניס קוד סאשן להתחבר לסאשן. לאחר שנכניס קוד ונלחץ על כפתור הקוד ישלח לשרת ונקבל תשובה. אם הקוד תקין נעבור למסך ההתחברות אם לא נציג הודעה למשתמש בהתאם. |  |
| ConnectingPanel | המחלקה יוצרת פאנל מסך התחברות לסאשן.  במחלקה זו ניצור מסך שבו נראה טקסט התחברות לסאשן ומידע על איך להתנתק מהסאשן אחרי שהתחברנו. נראה אנימציה של נקודות ולאחר 5 שניות המסך יעלם. |  |

## תיאור סביבת הפיתוח

שפת תכנות: שפת התכנות שבחרתי לעבוד איתה בפרויקט זה היא Python 3.7, בנוסף אעבוד עם SQL בשביל לתקשר עם הבסיס נתונים.

כלים לפיתוח ובדיקה:

* PyCharm: לסביבת עבודה נוחה לכתיבת קוד ובשביל Debugging
* Wireshark- לבדיקת תקשורת ומידע מוצפן שעובר ברשת
* DB Browser for SQLite: לבנות בסיס נתונים שאפשר להכניס לתוכו מידע שלנו
* CMD- להרצת פקודות ותכניות על המחשב
* File Explorer: לעבודה עם קבצים תמונות וכל דבר אחר שמאוכסן על המערכת קבצים במחשב

## תיאור האלגוריתמים המרכזיים בפרויקט:

סוגיה: כיצד לממש את החיבור peer2peer?

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | חיבור הלקוחות דרך השרת | המידע עובר דרך "גורם רשמי" , מבוקר, אמין  השרת אחראי על החיבור ומוודא את הצדדים | איטי – מידע עובר דרך שלישי  MITM – גורם נוסף מסתכל על המידע שעובר |
| 2 | חיבור ישיר בין הלקוחות | הרבה יותר מהיר  אין MITM | אין בקרה על המידע דרך גורם רשמי |

החלופה הנבחרת היא 2 – חיבור ישיר בין הלקוחות, מכיוון שבמערכת כזו נדרש חיבור מהיר וגם הגנה מפני MITM

סוגיה: איך יעבור המידע החיבור peer2peer כלל המידע: מהמקלדת, מהעכבר ותמונת המסך?

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | סוקט יחיד | ניהול פשוט | איטי – כל המידע עובר דרך stream יחיד  תקורה – נדרש למיין את המידע |
| 2 | סוקט לכל סוג מידע | מהירות – כל מידע לא תלוי במידע האחר  פחות תקורות – אין צורך למיין (מימוש ריבוב) | שימוש במספר רב של סוקטים  קשה יותר למימוש |
| 3 | Thread לכל סוקט שמהווה סוג מידע אחר | מהירות – כל מידע לא תלוי במידע האחר  פחות תקורות – אין צורך למיין (מימוש ריבוב)  אין עיכובים אם תהליכון 'תקוע' מאיזו סיבה | שימוש במספר רב של סוקטים  קשה יותר למימוש  קשה לניהול משאבים משותפים |
| 4 | Process לכל סוקט שמהווה סוג מידע אחר | הרבה יותר מהיר – כל תהליך רץ בנפרד לחלוטין, אין חלוקת משאבים | שימוש במספר רב של סוקטים  מימוש משאבים משופים (אם יידרש) הרבה יותר קשה |

החלופה הנבחרת היא 4 – Process לכל סוקט שמהווה סוג מידע אחר – במערכת שלי חשובה המהירות יותר מכל דבר אחר (שיתוף מסך) ואין צורך בשיתוף בין המידע של התמונה, עכבר והמקלדת

סוגיה: באיזה צורה מעבירים את המידע שהשתנה בתמונה?

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | העברת תמונה שלמה כל פרק זמן | הרבה יותר קל למימוש  קל לפירוק ובניית פרוטוקול לשליחות | כבד מאוד- לשלוח תמונה כל פרק זמן מסוים לפענח אותה ולהציג אותה זו פעולה מאוד כבדה שיכולה לאט את כל המערכת |
| 2 | העברת תמונה שלמה כל פעם שמזהים שינוי בה | פחות כבד- מתבצעת שליחות בתדירות יותר נמוכה (תלוי שינוי המסך) | עדיין שליחה של תמונה שלמה שזו פעולה מאוד כבדה על התומך והמבקש תמיכה |
| 3 | העברת רק את השינוי של המסך כל פעם שמזהים בו שינוי | הרבה פחות כבד- לא רק שתדירות השליחות של התמונה מתבצעת בקצב יותר איטי גם לא נשלחת תמונה שלמה ורק השינוי. | קשה למימוש- זיהוי שינוי במסך+פירוק השינוי מהתמונה ושליחה של השינוי היא פעולה קשה למימוש |

החלופה הנבחרת היא 3 – העברת רק את השינוי של המסך כל פעם שמזהים בו שינוי, מכיוון שבמערכת כזו נדרשת מהירות מעל הכל, אפילו אם יותר קשה לממש את העברת שינוי במסך.

סוגיה: באיזה פרוטוקול תקשורת להעביר את התמונה בין המקבל תמיכה והתומך?

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | שימוש בפרוטוקול UDP להעביר תמונה | המידע עובר הרבה יותר מהיר מפרוטוקול TCP | לא אמין- המידע שנשלח דרך פרוטוקול UDP נשלח לרשת ואין בקרה ובדיקה אם הוא מגיע או לא מגיע |
| 2 | שימוש בפרוטוקול TCP להעביר מידע תמונה | הקשר שנוצר בין השרת והלקוח אמין מוצפן ונבדק כאשר מגיע | איטי מאוד-המידע עובר הרבה יותר לאט מUDP מכיוון שצריך לעשות לחיצת יד משולשת וכל המידע נבדק. |

החלופה הרצויה היא 1 – שימוש בפרוטוקול UDP להעביר תמונה, מכיוון שבמערכת כזו נדרשת מהירות מעל הכל, אך החלופה הנבחרת היא 2 - שימוש בפרוטוקול TCP להעביר מידע תמונה. נבחר ב TCP ולא UDP מכיוון שבחרנו להעביר רק את השינוי של התמונה אם הוא נאבד בגלל UDP נוצר בעיה. מעבר של רק השינוי יהיה יותר יעיל מלהעביר תמונה שלמה רק בUDP לכן נבחר TCP

סוגיה: איך לחבר משתמש תומך ונתמך?

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | השרת בוחר לחבר את התומך ונתמך הראשונים אשר מתחברים למערכת (לפי סדר) | קל למימוש- כל פעם שיש לשרת תומך ונתמך במערכת אז הוא מחבר אותם  אמין- השרת בוחר איזה תומך מתחבר לנתמך, לכן הקשר שנוצר הוא אמין יותר | לא מתאים לצרכי הנתמך- מבקשי תמיכה העובדים במערכת כמעט תמיד מקבלים תמיכה ממישהו ספציפי שגם בתוך המערכת, לכן אם השרת מחבר את הנתמך אוטומטית לתומך פנוי זה לא עוזר לנתמך. |
| 2 | תומך מקבל קוד רנדומלי שתקף תמיד מהשרת והנתמך מכניס אותו בשביל להתחבר | מתאים לצרכי הנתמך- הנתמך יכול להכניס קוד (אשר קיבל מהתומך) בשביל להתחיל את הקשר לכן הוא בוחר למי להתחבר | לא אמין- מכיוון שהקוד ממשיך להיות תקף גם כאשר הקשר מתחיל\מתנתק, MITM יכול תמיד לנסות להתחבר במקום נתמך |
| 3 | תומך מקבל קוד רנדומלי שמוגבל בזמן מהשרת והנתמך מכניס אותו בשביל להתחבר | מתאים לצרכי הנתמך- הנתמך יכול להכניס קוד (אשר קיבל מהתומך) בשביל להתחיל את הקשר לכן הוא בוחר למי להתחבר    אמין – בגלל שהקוד התקבל מהשרת ומוגבל בזמן האמינות שלו תלויה רק בתומך ולמי הוא שולח את הקוד. לאחר שמתחבר נתמך אין אפשרות להמשיך להשתמש בקוד | קשה למימוש- בשרת צריך לממש מחולל קודים אשר ישמור כמה זמן מוגבל לכל קוד. דבר זה מסובך למימוש בצד השרת מבחינת שמירת ובדיקת הזמן והקוד שהוכנס. |

החלופה הנבחרת היא 3 – תומך מקבל קוד רנדומלי שמוגבל בזמן מהשרת והנתמך מכניס אותו בשביל להתחבר, אנחנו רוצים שהמערכת שלנו גם תתאים לצרכי המשתמשים וגם תהיה אמינה. חלופה זו מבצעת זאת לכן נבחר בה.

סוגיה: באיזה סוג הצפנה להעביר מידע ברשת?

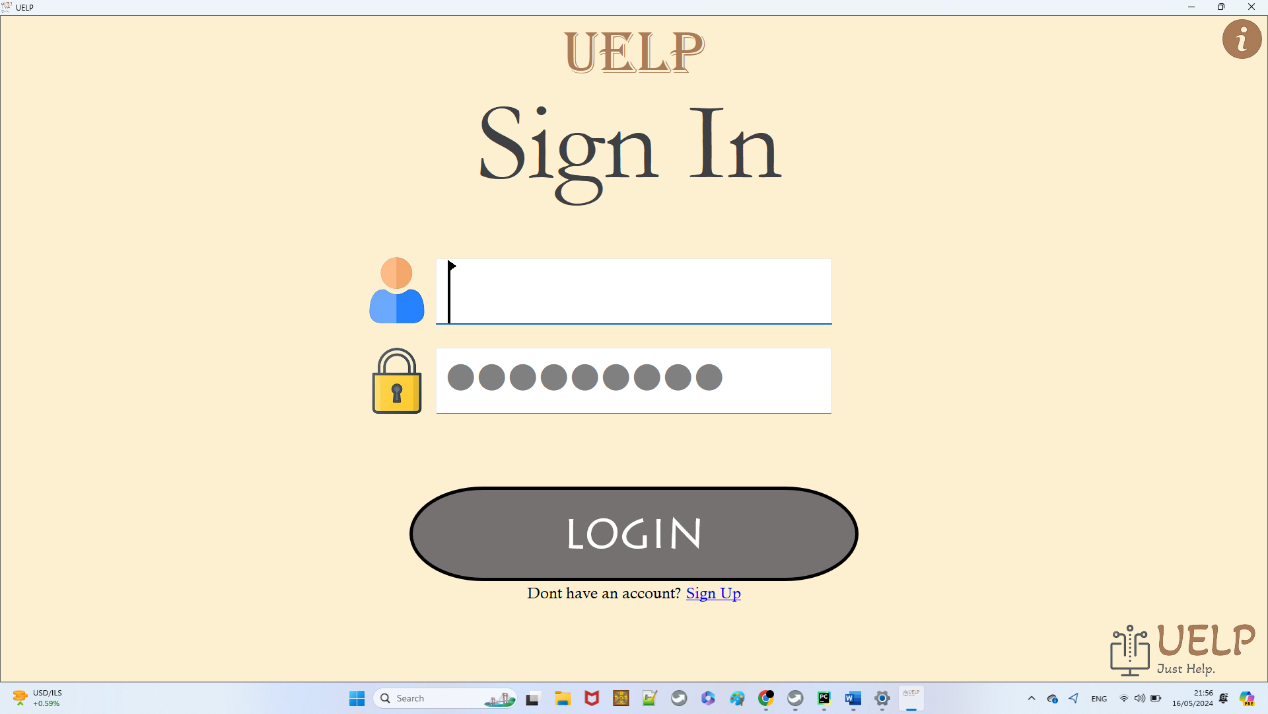
| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | הצפנה סימטרית | קל למימוש- הצפנה סימטרית לא קשה למימוש  מהירה- תהליך ההצפנה מהיר ולא מצריך הרבה זמן בשביל להצפין ולפענח מידע | בעיית המפתחות – הצפנה סימטרית תלויה במפתח משותף לנתמך ולתומך, בתוך הרשת אין איך להסכים על מפתח משותף בין המשתמשים בלי שיעבור באופן גלוי ברשת |
| 2 | הצפנה א-סימטרית | אין בעיית המפתחות-  הצפנה א-סימטרית עובדת על מפתח ציבורי ומפתח אישי, כך כל משתמש יכול להצפין למשתמש אחר מידע בעזרת המפתח הציבורי שלו ורק המשתמש שהמפתח ציבורי שלו יכול לפענח את המידע בעזרת המפתח האישי שלו.  בעזרת הצפנה זו אין בעיית מפתחות ואפשר להעביר מידע מוצפן בלי לעבוד עם מפתח משותף בין משתמשים. | איטי-  הצפנה א-סימטרית בדרך כלל משתמשת במפתחות ציבוריים ופרטיים מאוד מאוד ארוכים. בנוסף תהליך ההצפנה מאוד כבד לכן להצפין הרבה מידע יאט את המערכת |
| 3 | הצפנה היברידית | אין בעיית מפתחות – בהצפנה היברידית משתמשים בהצפנה א-סימטרית להעביר מפתח משותף בין התומך והנתמך בעזרת מפתח ציבורי ומפתח פרטי. לאחר מכן המשתמשים עוברים לתקשר עם הצפנה סימטרית שיותר מהיר ומשתמש במפתח משותף שגלוי רק להם | קשה יותר למימוש-  שימוש בשני שיטות הצפנה יכול לסבך את הקוד ויהיה קשה לממש |

החלופה הנבחרת היא 3 – הצפנה היברידית, מכיוון שהשימוש המשותף של שני שיטות ההצפנה עושה הבדל ענק במהירות ומביא את הטוב משני העולמות- הצפנה א-סימטרית פותר את בעיית המפתחות ויותר קל לעבוד עם הצפנה סימטרית כאשר יש לנו מפתח משותף הגלוי רק לתומך והנתמך.

## תיאור מסכי הפרויקט:

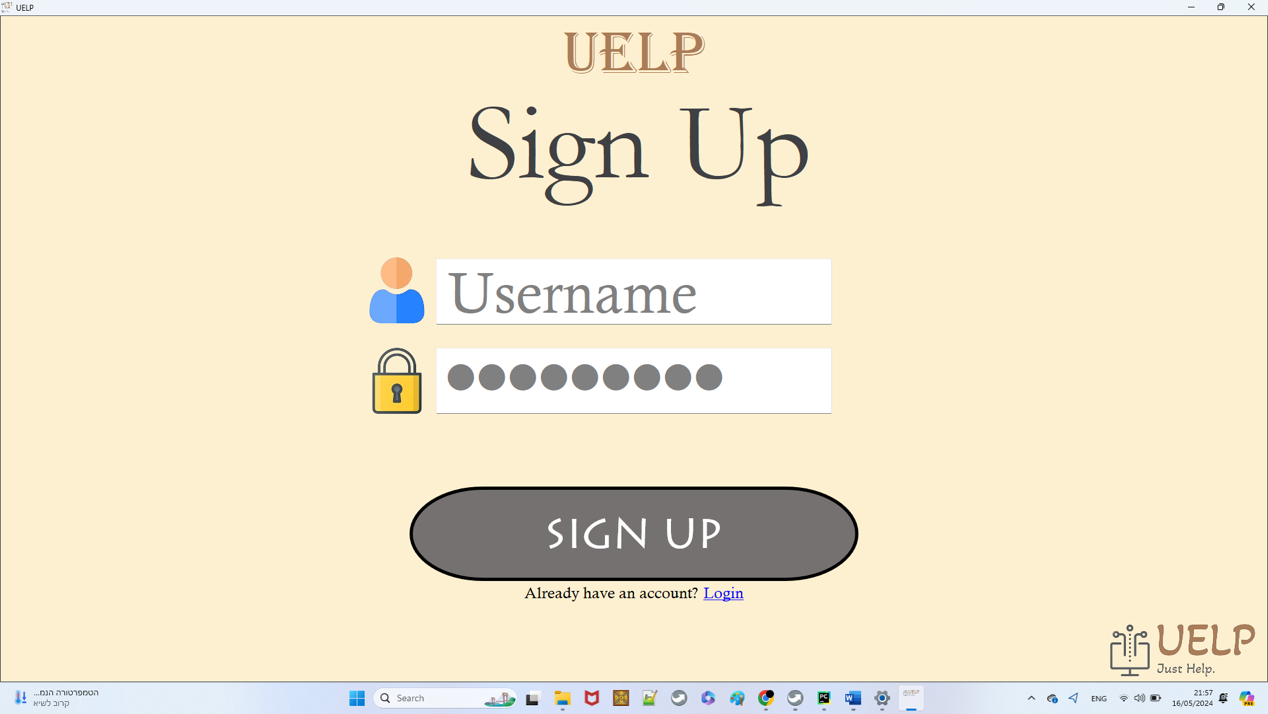
מסך 1: התחברות למערכת

מסך זה נועד בשביל להתחבר למערכת אם המשתמש רוצה להירשם למערכת הוא יכול ללחוץ על כפתור הההרשמה ולעבור למסך 2 מסך ההרשמה. מסך זה יהיה המסך הראשון שיופיע כשנכנסים למערכת. כאשר מכניסים שם משתמש וסיסמא השרת בודק אם הם קיימים במערכת ותקינים. לאחר שמשתמש מתחבר למערכת הוא עובר למסך 3: בחירת סוג משתמש ששם הוא בוחר אם הוא תומך או מבקש תמיכה.



מסך 2: הרשמה למערכת

מסך זה נועד בשביל להתחבר למערכת, אם המשתמש רוצה להירשם למערכת הוא יכול ללחוץ על כפתור ההתחברות ולעבור למסך 1 מסך ההתחברות. כאשר מכניסים שם משתמש וסיסמא השרת בודק אם הם יכולים להיות מוספים למערכת ותקינים. לאחר שמשתמש נרשם למערכת הוא עובר למסך 3: בחירת סוג משתמש ששם הוא בוחר אם הוא תומך או מבקש תמיכה.



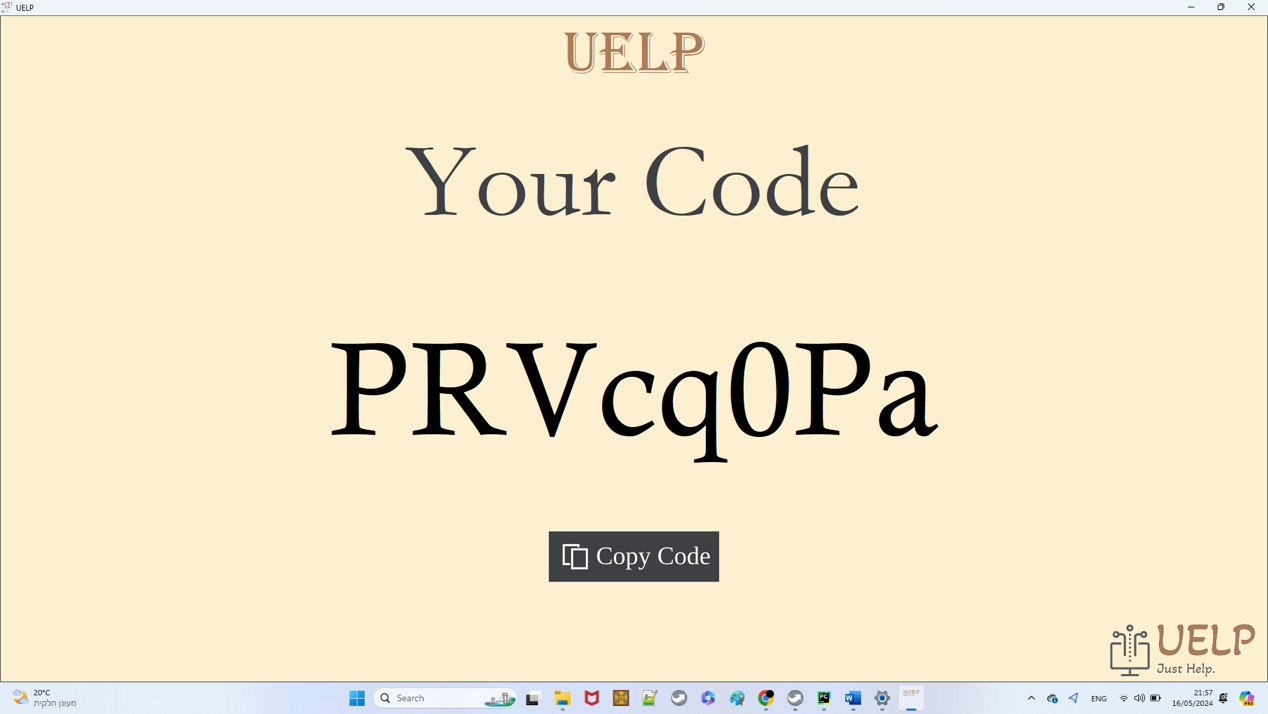
מסך 3: בחירת סוג משתמש

מסך זה יופיע למשתמש אחרי שהוא מתחבר למערכת. מטרת מסך זה היא שהמשתמש בוחר אם הוא רוצה להביא תמיכה למשתמשים אחרים או רוצה לקבל תמיכה ממשתמשים אחרים. השרת מאשר את המשתמש אם הוא רוצה להיות תומך לפי הBlacklist שלו בבסיס הנתונים. לאחר שהמשתמש בוחר סוג משתמש והשרת מאשר אותו, משתמשי תומכים יעברו למסך 3: קבלת קוד חדש ומשתמשי מבקשי תמיכה יעברו למסך 4: הכנסת קוד לפתיחת סאשן.



מסך 3: קבלת קוד חדש

מסך זה יופיע לאחר שמשתמש מאושר כתומך במערכת. מטרת מסך זה היא לפתוח קוד חדש על מנת שמבקש תמיכה יכניס אותו וכך יפתח סאשן. כאשר המשתמש מגיע למסך הזה, השרת יוצר קוד רנדומלי אשר מוגבל בזמן. לאחר קבלת הקוד המשתמש עובר למסך 5: התחברות לסאשן.

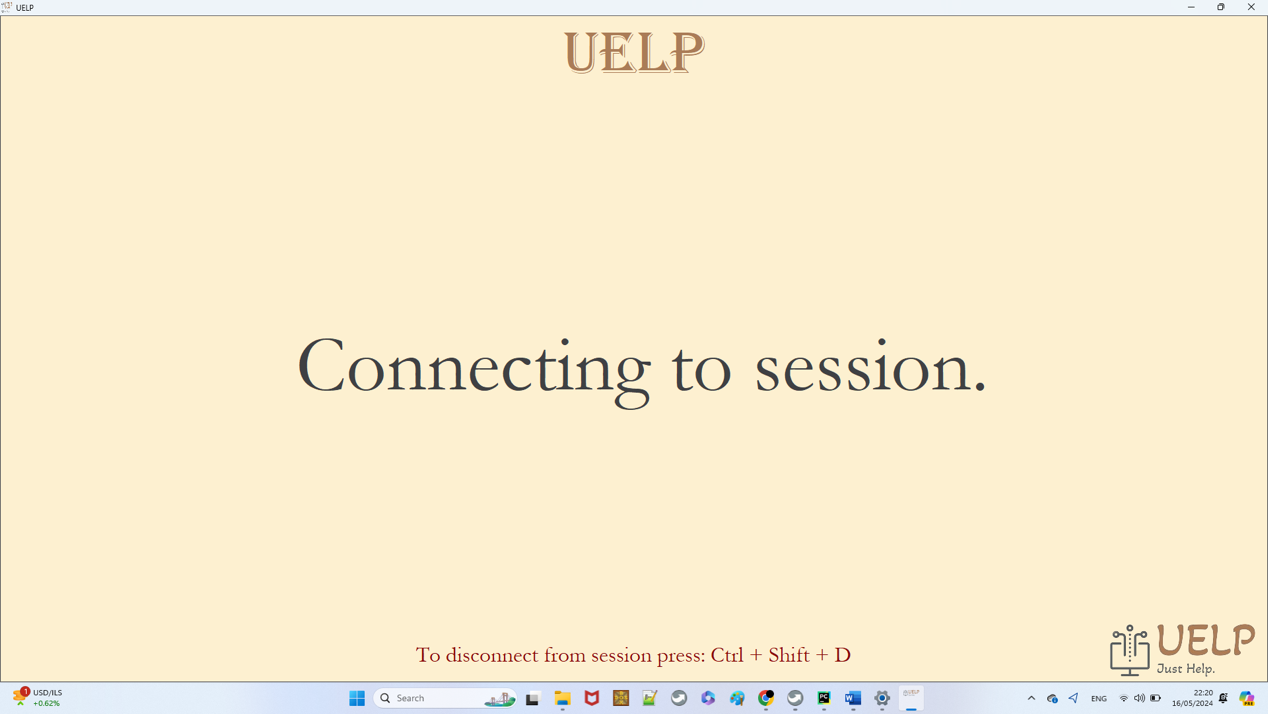


מסך 4: הכנסת קוד לפתיחת סאשן

מסך זה יופיע לאחר שמשתמש בחר ואושר כמבקש תמיכה. מטרת מסך זה היא להכניס קוד שהמשתמש קיבל מתומך אחר וכך להתחבר לסאשן שיצר התומך. הקוד שהכניס המבקש תמיכה נשלח לשרת והשרת בודק אם הקוד קיים והוא תקף, במידה וכן המשתמשים עוברים למסך 5: התחברות לסאשן.



מסך 5: התחברות לסאשן

תומך ומבקש תמיכה מגיעים למסך זה לאחר שהם מצטרפים לסאשן. מטרת מסך זה היא להראות למתחברים מידע לגבי הסאשן שהם מתחברים אליו כמו איך שהם מתנתקים מהסאשן תמיכה במידה ורוצים. לאחר 5 שניות במסך ההתחברות יעבור התומך למסך 6: מסך השיתוף, מסך זה הוא האחרון של המבקש תמיכה.

מסך 6: מסך השיתוף

תומך מגיע למסך זה כאשר המבקש תמיכה מצטרף לסאשן שלו לאחר שהכניס את הקוד שלו ועבר את מסך ההתחברות לסאשן. מטרת מסך זה הוא שהתומך יכול לראות את המסך מחשב של המבקש תמיכה ויכול לשלוט על העכבר והמקלדת של המבקש תמיכה. ממסך זה התומך מסיים את התמיכה לאחר שהמבקש תמיכה מבקש לעשות זאת או שהוא מבקש להתנתק מהמערכת.



## תיאור פרוטוקול התקשורת

שרת <-> לקוח

**\*כל ההודעות עוברות עם שדה אורך XXX בהתחלה**

|  | סוג הודעה | ממי למי? | שדות | דוגמא |
| --- | --- | --- | --- | --- |
| 1 | בניית הצפנה א-סימטרית: שליחת מפתח גלוי של שרת | ש -> ל | מפתח גלוי - 1234 | 1234 |
| 2 | בניית מפתח משותף להצפנה סימטרית: העברת מפתח פרטי מוצפן | ל -> ש | מפתח משותף (מוצפן) - 56894 | 56894 |

**מנקודה זו כל המידע העובר בהנחה מוצפן אך רשום כאילו הוא לא מוצפן**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | mac address | ל -> ש | mac address-  00-B0-D0-63-C2-26 | 0600-B0-D0-63-C2-26 |
| 4 | login/sign up | ל -> ש | opcode- 01/00  username\_len- 6  Itamar - username  Itamar6 - password | 016ItamarItamar6 |
| 5 | login/ sign up | ש -> ל | opcode- 01/00  0/1 – success/fail | 010 |
| 66 | user type | ל -> ש | 02 -opcode  0/1 – helper/assistance seeker | 020 |
| 7 | user type | ש -> ל | 02 -opcode  0/1 – success/fail | 020 |
| 8 | קבלת קוד חדש | ל -> ש | 03 -opcode | 03 |
| 9 | קבלת קוד חדש | ש -> ל | 03 -opcode  78634 – code | 0378634 |
| 10 | תוקף קוד תם | ש -> ל | 03 -opcode  expired-2 | 032 |
| 11 | הכנסת קוד | ל -> ש | 04 -opcode  43892 -code | 0443892 |
| 12 | הכנסת קוד | ש -> ל | 04 -opcode  0/1 – success/code invalid | 040 |
| 13 | חיבור חדש מול משתמש | ש -> ל | 05 -opcode  127.0.0.1 – ip of other user  userType- A | 05127.0.0.1A |

נתמך <-> תומך: עכבר (port:2001)

**\*כל ההודעות עוברות עם שדה אורך XXX בהתחלה כהנחה**

|  | סוג הודעה | ממי למי? | שדות | דוגמא |
| --- | --- | --- | --- | --- |
| 1 | בניית הצפנה א-סימטרית: שליחת מפתח גלוי של תומך | ת -> נ | מפתח גלוי - 1234 | 1234 |
| 2 | בניית מפתח משותף להצפנה סימטרית: העברת מפתח פרטי מוצפן | נ -> ת | מפתח משותף (מוצפן) - 56894 | 56894 |

**מנקודה זו כל המידע העובר בהנחה מוצפן אך רשום כאילו הוא לא מוצפן**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | העברת מיקום ולחיצת עכבר | ת -> נ | 1432 – x position of mouse  854 – y position of mouse  0/1/2/3/4/5/6 -left click/right click/ scroll up/ scroll down/ nothing/left click release/ right click release | 143208540 |

נתמך <-> תומך: מקלדת (port:2002)

**\*כל ההודעות עוברות עם שדה אורך XXX בהתחלה כהנחה**

|  | סוג הודעה | ממי למי? | שדות | דוגמא |
| --- | --- | --- | --- | --- |
| 1 | בניית הצפנה א-סימטרית: שליחת מפתח גלוי של תומך | ת -> נ | מפתח גלוי - 1234 | 1234 |
| 2 | בניית מפתח משותף להצפנה סימטרית: העברת מפתח פרטי מוצפן | נ -> ת | מפתח משותף (מוצפן) - 56894 | 56894 |

**מנקודה זו כל המידע העובר בהנחה מוצפן אך רשום כאילו הוא לא מוצפן**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | העברת לחיצת מקלדת | ת -> נ | opcode- 01  'a'-key | 01ord('a') |
| 4 | העברת עזיבת לחיצה על מקלדת | ת -> נ | 02 -opcode  'a' -key | 02ord('a') |
| 5 | העברת עזיבת מקשי התנתקות מסאשן | ת -> נ | 02 -opcode  9999999 | 029999999 |

נתמך <-> תומך: מסך (port:2003)

**\*כל ההודעות עוברות עם שדה אורך XXX בהתחלה כהנחה**

|  | סוג הודעה | ממי למי? | שדות | דוגמא |
| --- | --- | --- | --- | --- |
| 1 | בניית הצפנה א-סימטרית: שליחת מפתח גלוי של תומך | ת -> נ | מפתח גלוי - 1234 | 1234 |
| 2 | בניית מפתח משותף להצפנה סימטרית: העברת מפתח פרטי מוצפן | נ -> ת | מפתח משותף (מוצפן) - 56894 | 56894 |

**מנקודה זו כל המידע העובר בהנחה מוצפן אך רשום כאילו הוא לא מוצפן**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | העברת אזור שינוי ושינוי במסך | נ -> ת | 01-opcode  0300 -left  0400 -top  1200 -right  1000 -bottom  1234- image data size  Image data | 0103000400120010001234data |
| 4 | העברת תמונת מסך שלמה | נ -> ת | 02 -opcode  1234- image data size  Image data | 021234data |

## תיאור מבני הנתונים

משתנים מקומיים:

| שם | טיפוס | שימוש | דוגמה |
| --- | --- | --- | --- |
| open\_client | dictionary(sockets: (IP, encryption\_object) | קישור בין sockets פתוחים לבין ה IP של ה socket והעצם הצפנה שבעזרתו מצפינים את המידע  נמצא בשרת | (socket object: '192.3.4.203: encryption\_object') |
| users | dictionary(IP:(mac address, type\_user) | קישור בין כתובת ip כתובת mac של לקוח וסוג משתמש הלקוח | ('192.3.4.203': '00-B0-D0-63-C2-26': 'H') |
| commands | dictionary(opcode:function) | קישור בין אופקוד לפונקציה שקוראים לה מהאופקוד הזה | (("00": handle\_login\_ans |
| currScreen | עצם PIL של תמונת מסך נוכחי | לשמור את המסך הנוכחי להדבקה עליו ותצוגה |  |

**מסד נתונים:**

לשרת יהיה גישה למסד נתונים אחד בשם Uelp\_db

במסד נתונים זה תהיה טבלה בשם users עם המידע:

עמודה 1 (username): שם משתמש של משתמש. ((primary key

עמודה 2 (:(password hash של הסיסמא של השם משתמש

|  |  |
| --- | --- |
| Password | Username |
| 4b4b0f17da04db1de6d24fade424adb13c41a802bae8fdb723df0fc43391b3cb | Itamarnahum234 |
| ffd8bc653cf25ed962af9af60aaea2d703213316feea067e40b15c847cb73fec | Moshbenaro27272 |
| a92f49d0461deffcf4b8b02149e591fa663d6052cec4fd714064d2beadd37ca7 | Merrygeva2024 |
| b183af27552242fc4addf8fa6498ff9d62d7ba6d82a4d7f22b7c848fc4f88043 | Kerenkuper023 |

במסד נתונים תהיה טבלה נוספת בשם macs עם הhash של הכתובת הפיזית של כל המשתמשים שחסומים מלהיות תומך במערכת:

עמודה 1 ( hash :(mac addressשל הכתובת הפיזית של user (primary key)

|  |
| --- |
| Mac addresses |
| 8c523a89ff59046ef5c827f3f3575212944f55a99992fe14824a085ea775d098 |
| d90e45e4258d7fae3e49a24b940a7ea46860621fecd5b948bb48971ae9f52152 |
| b2ca5ab19a63d223c36c9d81c36ea16d34a461259b8becf1c56ca834284aa5f8 |

## סקירת חולשות והאיומים

|  | חולשה | סכנות | פתרון |
| --- | --- | --- | --- |
| 1 | SQL Injection | sql injection היא מתקפה נפוצה מאוד שמשתמשים בה להשיג מידע רגיש מבסיסי נתונים ולגשת למידע שלא אמורים לגשת.  עושים זאת על ידי הכנסת קוד sql בתוך מקומות שהמשתמש חושב שמריצים את הinput שלו כחיפוש בבסיס נתונים | כתיבת שאילתות נכונות כשניגשים לטבלאות בבסיס נתונים.  כך שלא מתאפשר הכנסת קוד זדוני והשגת טבלאות שלא אמורות להיות גישה אליהם |
| 2 | גניבת מידע שמור | מידע ששמור לאורך זמן יכול להיגנב אם נשמר באופן פומבי | שמירת מידע כhash מגן עליו כך שאי אפשר להפוך פונקציית hash ורק להשוות אליה.  אצלי בפרויקט כל מידע רגיש וחשוב שמור עם hash |
| 3 | Man in the middle | מתקפה זו פועלת כך שמישהו ברשת מאזין לשיחה אשר מתקיימת בין שתי ישויות וכך יכול להיחשף למידע רגיש או אפילו לשנות את אופן השיחה ולהשפיע על הישויות בעזרת התחזות לאחד מהן | אצלי בפרויקט כל מידע שמועבר ברשת מוצפן. תחילה בונים מפתח סימטרי משותף בעזרת הצפנה א-סימטרית ולאחר מכן כל מידע שעובר ברשת מוצפן בעזרת המפתח המשותף. |
| 4 | פרוטוקולי תקשורת | רשת אינטרנט נחשבת לא בטוחה ובשביל שמידע יגיע ליעדו צריך להשתמש בפרוטוקולי תקשורת בטוחים וידועים | בפרויקט שלי אני משתמש בפרוטוקול תקשורת tcp שמבטיח שההודעות יגיע בסדר הנכון ולישות הנכונה ברשת. דבר זה נעשה על ידי ה Three way Handshake שהפרוטוקול מבצע בעת חיבור ועל ידי הchecksum ובעזרת הצפנה |

# פרק ה' - UELP - הקוד

יכולת 1: קבלת תמונת המסך ויצירת תמונת שינוי מהתמונת מסך החדשה והנוכחית.

ביכולת זו אנחנו שומרים כל פעם את תמונת המסך הנוכחי, אנחנו מקבלים מחדש את המסך ובודקים אם התקיים בו שינוי, במידה וכן נבדוק כמה המסך השתנה ונשלח את התמונה החדשה או רק את תמונת השינוי שאפשר להדביק על התמונה הנוכחית למבקש תמיכה.

def main\_AS\_screen(otherIP):  
 *"""  
 Main Assistance Seeker screen, get new screenshot of screen check if there is difference and  
 send the difference or the full new screenshot accordingly  
 :param otherIP: ip of Helper server to send images too  
 """* pygame.init()  
 screen\_info = pygame.display.Info()  
 screen\_width, screen\_height = screen\_info.current\_w, screen\_info.current\_h  
 recv\_q = queue.Queue()  
 port = 2003  
  
 client = Client\_comm(otherIP, port, recv\_q)  
  
 *# get first screenshot and compress the image* currScreen = ImageGrab.grab()  
 currScreen\_bytes = currScreen.tobytes()  
 currScreen\_bytes = zlib.compress(currScreen\_bytes)  
  
 *# wait for client to finish key exchange then send the first image* while not client.exchangeStatus():  
 pass  
  
 *# send first screenshot to Helper based on protocol* client.sendImage(AssistanceSeeker\_protocol.pack\_full\_image(str(len(currScreen\_bytes))), currScreen\_bytes)  
  
 while True:  
 *# get new screenshot* newScreen = ImageGrab.grab()  
 diffImage, diff\_bbox = getChanged(newScreen, currScreen)  
 if diffImage is not None:  
 changed\_precentage = getChangedPrecentage(diff\_bbox, screen\_height \* screen\_width)  
 *# if Image is more then 75 percent different, send the entire Image and not the difference* if changed\_precentage >= 75:  
 newScreen\_bytes = newScreen.tobytes()  
 newScreen\_bytes = zlib.compress(newScreen\_bytes)  
 *# send full screenshot to Helper based on protocol* client.sendImage(AssistanceSeeker\_protocol.pack\_full\_image(str(len(newScreen\_bytes))),  
 newScreen\_bytes)  
  
 else:  
 *# get the bounding box values of the difference Image* top, left, bottom, right = diff\_bbox  
  
 diffBytes = diffImage.tobytes()  
 diffBytes = zlib.compress(diffBytes)  
 *# build difference Image header based on protocol* msg = AssistanceSeeker\_protocol.pack\_part\_image(str(top), str(left), str(bottom), str(right),  
 str(len(diffBytes)))  
  
 *# send difference Image and Image header to Helper* client.sendImage(msg, diffBytes)  
  
 *# update currScreen as newScreen* currScreen = newScreen

def getChanged(newScreen: ImageGrab, currScreen: ImageGrab):  
 *"""  
 Function checks if there is a difference between two images, if there is it returns the difference Image  
 (Image that represents the differnce between the two Images) and the bounding box of the difference Image  
 :param newScreen: the ImageGrab image of the new screen that might differ from the currScreen  
 :param currScreen: the ImageGrab image of the current screen  
 :return: if the Images don't differ return None otherwise return the diff Image and the bounding box of this Image  
 """  
 # Check if the images are equal* diff = ImageChops.difference(currScreen, newScreen)  
 bbox = diff.getbbox()  
  
 *# Images are identical* if bbox is None:  
 return None, None  
  
 *# Get the region of the difference* diff\_region = newScreen.crop(bbox)  
  
 return diff\_region, bbox  
  
  
def getChangedPrecentage(bbox, total\_pixels: int) -> float:  
 *"""  
 Function checks how much percentage Image changed since the last Image based on bounding box of diffImage  
 :param bbox: The bounding box of the difference Image on the screen  
 :param total\_pixels: the total amount of pixels on screen  
 :return: percentage representing how much the Image has changed based on the difference Image  
 """  
 # get loc of diff Image* top, left, bottom, right = bbox  
  
 *# calculate the percentage of difference based on the pixel amount of the difference in total pixels of screen* diff\_pixels = (bottom - top) \* (right - left)  
 percentage\_difference = (diff\_pixels / total\_pixels) \* 100  
  
 return percentage\_difference

יכולת 2: קבלת תמונות מסך מהמבקש תמיכה והצגה שלהם על המסך

ביכולת זו אנחנו מקבלים תמונות מסך מהמבקש תמיכה. להתחיל אנחנו מקבלים תמונת מסך מלאה ראשונה. לאחר מכן נקבל או תמונות מלאות או תמונות שינוי, אם נקבל תמונות שינוי נדביק אותה על התמונה הנוכחית ואם תמונה מלאה נציג אותה במקום. נעדכן את התמונה הנוכחית בהתאם.

*# global current PIL image on screen*currScreen = None  
  
  
def build\_Screen(top: int, left: int, right: int, bottom: int, img\_bytes: bytes):  
 *"""  
 Function gets the location and bytes of the image to paste and pastes it on current screen  
 :param top: top value of Image  
 :param left: left value of Image  
 :param right: right value of Image  
 :param bottom: bottom value of Image  
 :param img\_bytes: bytes of Image  
 """  
 # location to paste Image* paste\_loc = (top, left)  
  
 *# Create a PIL Image object from the bytes of the image bytes* diff\_img = Image.frombytes('RGB', (bottom - top, right - left), img\_bytes)  
  
 *# Paste the diff image on the current screen* currScreen.paste(diff\_img, paste\_loc)  
  
  
def main\_Helper\_screen(otherIP):  
 *"""  
 Main Helper screen, get images from Assistance Seeker client and paste  
 the images based on if are difference or full Images accordingly  
 :param otherIP: ip of Assistance Seeker client getting images from  
 """* recv\_q = queue.Queue()  
 port = 2003  
 global currScreen  
 *# create new server that only gets messages from otherIP with port and recv\_q gotten* server = Server\_comm(recv\_q, port, otherIP)  
  
 *# create pygame screen and save width and height of screen* pygame.init()  
 screen\_info = pygame.display.Info()  
 screen\_width, screen\_height = screen\_info.current\_w, screen\_info.current\_h  
  
 *# wait until gotten first image from Assistance seeker to create pygame screen* while True:  
 if not recv\_q.empty():  
 screen = pygame.display.set\_mode((screen\_width, screen\_height), pygame.FULLSCREEN)  
 break  
  
 while True:  
 *# get data from client* if not recv\_q.empty():  
 data, ip = recv\_q.get()  
 if data == "disconnect":  
 pygame.quit()  
 break  
  
 *# if amount of params in data is 5, we got a part image, take out params decompress and  
 # build\_screen based on imageData gotten* if len(data) == 5 and currScreen:  
 top = data[0]  
 left = data[1]  
 bottom = data[2]  
 right = data[3]  
 diff\_bytes = data[4]  
 diff\_bytes = zlib.decompress(diff\_bytes)  
  
 top, left, right, bottom = int(top), int(left), int(right), int(bottom)  
  
 *# call \_build\_screen with all params of image to paste this image on the current screen* build\_Screen(top, left, right, bottom, bytes(diff\_bytes))  
 *# if only one param we got a full Image, decompress and build it as current screen* elif len(data) == 1:  
 newScreen\_bytes = data[0]  
 newScreen\_bytes = zlib.decompress(newScreen\_bytes)  
 *# Create a PIL Image object from the bytes of the image bytes* currScreen = Image.frombytes('RGB', (screen\_width, screen\_height),  
 bytes(newScreen\_bytes))  
  
 *# if there is currently a valid PIL image object as current screen show it to user using pygame* if currScreen is not None:  
 *# transform PIL image object to pygame image* currScreen\_bytes = currScreen.tobytes()  
 pygame\_currScreen = pygame.image.frombuffer(currScreen\_bytes,  
 (screen\_width, screen\_height), "RGB")  
 *# show image* screen.blit(pygame\_currScreen, (0, 0))  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 break

יכולת 3: יצירת קןדי סאשן ובדיקה ושמירה עליהם

ביכולת זו יש לנו מחלקה השומרת על כל קודי הסאשנים הקיימים כעת. אנו בודקים כל הזמן אם הקוד נהיה פג תוקף, במידה וכן נודיע למשתמש בלוגיקה ונעביר לה את זה בעזרת תור ובנוסף נבדוק שקוד שאנו מוסיפים לא קיים במערכת.

*# class to work with session codes of sessions valid in server*class Session\_codes:  
 def \_\_init\_\_(self, expired\_q):  
 self.expired\_q = expired\_q  
 self.codes = {}  
 self.cleanseTime = 180  
 threading.Thread(target=self.\_remove\_expired).start()  
  
 def \_remove\_expired(self):  
 *"""  
 Function checks dictionary of codes every two seconds, if there are codes who expired remove from dic and add to  
 expired codes queue  
 """* while True:  
 currTime = time.time()  
 expired\_ips = []  
  
 for ip in self.codes.keys():  
 createTime = self.codes[ip][1]  
 if currTime - createTime > self.cleanseTime:  
 expired\_ips.append(ip)  
 self.expired\_q.put(ip)  
  
 *# remove expired\_ips from dictionary* for ip in expired\_ips:  
 del self.codes[ip]  
  
 time.sleep(2)  
  
 def checkCode(self, code: str) -> bool:  
 *"""  
 Function checks if code gotten exists in dictionary of all codes  
 :param code: code to check if in dic  
 :return: True if code in codes dictionary, False otherwise  
 """* onlyCodes = [codeInfo[0] for codeInfo in self.codes.values()]  
 return code in onlyCodes  
  
 def createCode(self, ip: str) -> str:  
 *"""  
 Function creates a random string code that dosent exist in all codes and adds it to dic  
 :param ip: ip of user whose code was created for (saved in dic with code)  
 :return: code created  
 """  
 # getting random code and checking if exists* while True:  
 newCode = base64.b64encode(secrets.token\_bytes(6)).decode()  
 if not self.checkCode(newCode):  
 break  
  
 self.codes[ip] = (newCode, time.time())  
 return newCode  
  
 def ip\_from\_code(self, code: str) -> str:  
 *"""  
 Function returns the ip of a user from a code gotten  
 :param code: code of session that a user created  
 :return: ip of user that created the session code  
 """* ip = None  
 for ip\_dic, code\_time in self.codes.items():  
 if code == code\_time[0]:  
 ip = ip\_dic  
 break  
 return ip

יכולת 4: הפעלת פרוססים שונים לתומך ולמבקש תמיכה להתחלת הסאשן

ביכולת זו נסתכל על סוגיית ההפעלת הסאשן מהלוגיקה הראשית של קליינט של השרת לתומך או נתמך. נתנתק מהשרת הראשי ונפעיל את הפרוססים המתאימים לפי סוג משתמש, ניצור תור הודעות שיעבור בין הלוגיקה הראשית לעצם הבקרה על המקלדת כך המקלדת תוכך להודיע ללקוח מתי להרוג את כל הפרוססים לפי בקשות סיום סאשן מהמקלדת.

def handle\_conData(params):  
 *"""  
 Function gets connection data of other user to connect too and starts process for mouse keyboard and screen  
 accordingly to user type  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* otherIP = params[0]  
 my\_user\_Type = params[1]  
  
 if my\_user\_Type:  
 *# queue for comm between keyboard and client to send msgs when keyboard detects to disconnect from session* close\_queue = multiprocessing.Queue()  
 *# create proccesss for keyboard screen and mouse as both a Helper and Assistance Seeker accordingly  
 # if a user is a Helper send to graphic connecting\_session to switch to connecting session screen* if my\_user\_Type == "H":  
 mouse = multiprocessing.Process(target=Helper\_logic.main\_Helper, args=(otherIP, 2001, None,))  
 keyboard = multiprocessing.Process(target=Helper\_logic.main\_Helper, args=(otherIP, 2002, close\_queue,))  
 screen = multiprocessing.Process(target=Helper\_screen\_logic.main\_Helper\_screen, args=(otherIP,))  
 wx.CallAfter(pub.sendMessage, "connecting\_session")  
 elif my\_user\_Type == "A":  
 mouse = multiprocessing.Process(target=AssistanceSeeker\_mouse\_logic.main\_AS\_mouse, args=(otherIP,))  
 keyboard = multiprocessing.Process(target=AssistanceSeeker\_keyboard\_logic.main\_AS\_keyboard,  
 args=(otherIP, close\_queue,))  
 screen = multiprocessing.Process(target=AssistanceSeeker\_screen\_logic.main\_AS\_screen, args=(otherIP,))  
  
 *# close client to main server (not needed anymore as connection switches to P2P)* client.close()  
  
 *# wait five seconds before starting mouse screen and keyboard process for connecting to session screen  
 # to finish in graphics* time.sleep(5)  
 mouse.start()  
 keyboard.start()  
 screen.start()  
  
 *# call function for when user asks session to close* check\_closed(close\_queue, mouse, screen, keyboard)  
  
  
def check\_closed(close\_queue, mouse, screen, keyboard):  
 *"""  
 Function gets data from close\_queue when data gotten is close  
 (keyboard process tells the client to terminate all processes to end session)  
 terminate all running processes and leave loop  
 :param close\_queue: multiprocessing queue for comm between client and keyboard processes  
 :param mouse: running process of mouse control  
 :param screen: running process of screen control  
 :param keyboard: running process of keyboard control  
 """* while True:  
 data = close\_queue.get()  
 if data == "close":  
 mouse.terminate()  
 screen.terminate()  
 keyboard.terminate()  
 break

יכולת 5: בקרה על מקלדת של התומך והעברת מידע מהמקלדת

בפעולות אלו אנחנו מקשיבים למקלדת של התומך, אנחנו נבדוק מתי המקלדת נלחצת או נעזבת ונשלח בהתאמה למבקש תמיכה. נבדוק אן נלחצה קומבינציית הניתוק מהסאשן אם כן נשלח אופקוד של עזיבה של מקשים אלו למבקש תמיכה שידע לעזוב את מקשי הקומבינציה ולהתנתק מהסאשן. נשלח הודעה בתור ללוגיקה של הלקוח להרוג את כל הפרוססים של השליטה ונסיים את הבקרה על המקלדת.

*# class to monitor keyboard*class Keyboard\_monitor:  
 def \_\_init\_\_(self, server, clientIP: str, close\_queue):  
 *"""  
 Builder function creates new Keyboard\_monitor object with the vars gotten and calls function \_monitor\_keyboard  
 :param server: server to send keyboard data through  
 :param clientIP: client ip to send keyboard data too  
 """* self.server = server  
 self.clientIP = clientIP  
 *# values for keys who do not have values in pynput* self.special\_keys\_mapping = {Key.ctrl\_l: 1114112,  
 Key.ctrl\_r: 1114113,  
 Key.shift\_l: 1114114,  
 Key.shift\_r: 1114115,  
 Key.alt\_l: 1114116,  
 Key.alt\_r: 1114117,  
 Key.cmd: 1114118,  
 Key.cmd\_l: 1114119,  
 Key.cmd\_r: 1114120,  
 Key.enter: 1114121,  
 Key.backspace: 1114122,  
 Key.tab: 1114123,  
 Key.space: 1114124,  
 Key.delete: 1114125,  
 Key.esc: 1114126,  
 Key.f1: 1114127,  
 Key.f2: 1114128,  
 Key.f3: 1114129,  
 Key.f4: 1114130,  
 Key.f5: 1114131,  
 Key.f6: 1114132,  
 Key.f7: 1114133,  
 Key.f8: 1114134,  
 Key.f9: 1114135,  
 Key.f10: 1114136,  
 Key.f11: 1114137,  
 Key.f12: 1114138,  
 Key.home: 1114139,  
 Key.end: 1114140,  
 Key.page\_up: 1114141,  
 Key.page\_down: 1114142,  
 Key.insert: 1114143,  
 Key.menu: 1114144,  
 Key.caps\_lock: 1114145,  
 Key.num\_lock: 1114146,  
 Key.scroll\_lock: 1114147,  
 Key.pause: 1114148,  
 Key.print\_screen: 1114149,  
 Key.shift: 1114150,  
 Key.ctrl: 1114151,  
 Key.alt: 1114152,  
 Key.up: 1114153,  
 Key.down: 1114154,  
 Key.left: 1114155,  
 Key.right: 1114156}  
 self.close\_queue = close\_queue  
 self.listener = None  
 self.\_monitor\_keyboard()  
  
 def pressed\_end(self):  
 *"""  
 Function is called when combination Shift + Ctrl + d is pressed, send data to client based on protocol and  
 release Shift Ctrl and d  
 """  
 # release shift ctrl and d* self.server.send(self.clientIP, Helper\_protocol.pack\_key\_release("9999999"))  
 self.stop\_listening()  
  
 *# wait 0.1 seconds for client to recieve end msg then place close in close\_queue to let client know  
 # to terminate all proccesses* time.sleep(0.1)  
 self.close\_queue.put("close")  
  
 def \_monitor\_keyboard(self):  
 *"""  
 Function monitors and listens to keyboard clicks and releases and calls functions accordingly  
 """* self.listener = keyboard.Listener(on\_press=self.\_on\_press, on\_release=self.\_on\_release)  
 *# add hotkey for end session combination and function for it* Keyboard.add\_hotkey("ctrl+shift+d", self.pressed\_end)  
 self.listener.start()  
  
 def stop\_listening(self):  
 *"""  
 Stops the keyboard listener.  
 """* if self.listener.is\_alive():  
 self.listener.stop()  
  
 def \_on\_press(self, key):  
 *"""  
 Function is called when key is pressed, and checks if key is a special key or not and sends value of  
 key based on protocol and special keys mapping  
 :param key: key pressed on keyboard  
 """* if key in self.special\_keys\_mapping.keys():  
 msg = str(self.special\_keys\_mapping[key])  
 else:  
 msg = str(ord(key.char))  
 msg = Helper\_protocol.pack\_key\_click(msg)  
  
 self.server.send(self.clientIP, msg)  
  
 def \_on\_release(self, key):  
 *"""  
 Function is called when key is released and checks if key is in special keys, sends the value of key based on  
 protocol and special keys mapping  
 :param key: key released on keyboard  
 """* if key in self.special\_keys\_mapping.keys():  
 msg = str(self.special\_keys\_mapping[key])  
 else:  
 msg = str(ord(key.char))  
 msg = Helper\_protocol.pack\_key\_release(msg)  
 self.server.send(self.clientIP, msg)

יכולת 6: קבלה של מידע המקלדת מהתומך וביצועו על מקלדת של המחשב

בפעולות אלו אנו מקבלים מידע על פעולות מקלדת מהתומך. אנו נבצע את פעולות אלו על המקלדת שלנו ונבדוק בו זמנית אם נלחץ קומבינציית העזיבת סאשן, אם כן נעביר בתור הודעות בין הלוגיקה של קליינט לפרוסס מקלדת של המקש תמיכה הודעה אשר אומרת שהסאשן נגמר וצריך להרוג את כל הפרוססים של השליטה.

def main\_AS\_keyboard(otherIP, close\_queue):  
 *"""  
 Function controls keyboard click and releases gotten from Helper, unpack data and register  
 key click on computer keyboard  
 :param otherIP: ip of server of Keyboard helper   
 :param close\_queue: multiprocessing queue between main client and keyboard process to  
 let main client know when to close session  
 """* port = 2002  
 recv\_q = queue.Queue()  
 client = Client\_comm(otherIP, port, recv\_q)  
  
 *# keyboard controller object* keyboard\_cont = keyboard.Controller()  
  
 *# values for keys who do not have values in pynput* special\_keys\_mapping = {  
 1114112: Key.ctrl\_l,  
 1114113: Key.ctrl\_r,  
 1114114: Key.shift\_l,  
 1114115: Key.shift\_r,  
 1114116: Key.alt\_l,  
 1114117: Key.alt\_r,  
 1114118: Key.cmd,  
 1114119: Key.cmd\_l,  
 1114120: Key.cmd\_r,  
 1114121: Key.enter,  
 1114122: Key.backspace,  
 1114123: Key.tab,  
 1114124: Key.space,  
 1114125: Key.delete,  
 1114126: Key.esc,  
 1114127: Key.f1,  
 1114128: Key.f2,  
 1114129: Key.f3,  
 1114130: Key.f4,  
 1114131: Key.f5,  
 1114132: Key.f6,  
 1114133: Key.f7,  
 1114134: Key.f8,  
 1114135: Key.f9,  
 1114136: Key.f10,  
 1114137: Key.f11,  
 1114138: Key.f12,  
 1114139: Key.home,  
 1114140: Key.end,  
 1114141: Key.page\_up,  
 1114142: Key.page\_down,  
 1114143: Key.insert,  
 1114144: Key.menu,  
 1114145: Key.caps\_lock,  
 1114146: Key.num\_lock,  
 1114147: Key.scroll\_lock,  
 1114148: Key.pause,  
 1114149: Key.print\_screen,  
 1114150: Key.shift,  
 1114151: Key.ctrl,  
 1114152: Key.alt,  
 1114153: Key.up,  
 1114154: Key.down,  
 1114155: Key.left,  
 1114156: Key.right}  
  
 *# start thread for checking close session combo and comm with main client* closed = threading.Thread(target=check\_close, args=(close\_queue, client,))  
 closed.start()  
  
 while True:  
 data = recv\_q.get()  
 if data != "close":  
 opcode, key\_val = protocol.unpackData(data)  
 key\_val = int(key\_val)  
 *# gotten end session opcode from Helper, release end session combo and end loop* if key\_val == 9999999:  
 keyboard\_cont.release(Key.shift)  
 keyboard\_cont.release(Key.ctrl)  
 keyboard\_cont.release("d")  
 print("released end session combo")  
 break  
 *# check if value gotten is registered in special keys and translate to key accordingly* if key\_val in special\_keys\_mapping.keys():  
 key = special\_keys\_mapping[key\_val]  
 else:  
 *# add value for keys gotten between 0 and 27 as  
 # these keys are (Ctrl + key) and are not registered correctly in pynput* if 0 < key\_val < 27:  
 key\_val += 96  
 key = chr(key\_val)  
  
 *# check if key is released or clicked and change computer keyboard accordingly* if opcode == "01":  
 keyboard\_cont.press(key)  
 print("pressed " + key)  
 elif opcode == "02":  
 keyboard\_cont.release(key)  
 print("released " + key)  
 else:  
 break

def check\_close(close\_queue, client):  
 *"""  
 Function waits until detected close session combo closes client and puts close in close queue  
 :param close\_queue: multiprocessing queue between main client and keyboard process to  
 let main client know when to close session  
 :param client: client object to comm with server  
 """* while True:  
 if Keyboard.is\_pressed("shift+ctrl+d"):  
 close\_queue.put("close")  
 client.close()  
 break

יכולת 7: בקרה על עכבר של התומך והעברת מידע מעכבר

בפעולות אלו אנחנו מקשיבים לעכבר של התומך, אנחנו נבדוק מתי העכבר נלחץ או נעזב ואיזה סוג לחיצה זו, מתי הוא זז או משתמש בגלגלת ונשלח בהתאמה למבקש תמיכה.

*# class to monitor mouse*class Mouse\_monitor:  
 def \_\_init\_\_(self, server, clientIP: str):  
 *"""  
 Builder function creates new Mouse\_monitor object to monitor mouse functions  
 :param server: server to send through data about mouse  
 :param clientIP: ip of client to send mouse data too  
 """* self.server = server  
 self.clientIP = clientIP  
 self.listener = None  
  
 self.\_monitor\_mouse()  
  
 def \_monitor\_mouse(self):  
 *"""  
 Function creates a new listener for the mouse to listen for scroll clicks and movement  
 """* self.listener = pynput.mouse.Listener(on\_move=self.\_on\_move, on\_click=self.\_on\_click,  
 on\_scroll=self.\_on\_scroll)  
 self.listener.start()  
  
 def stop\_listening(self):  
 *"""  
 Stops the mouse listener.  
 """* if self.listener:  
 self.listener.stop()  
  
 def \_on\_move(self, x: int, y: int):  
 *"""  
 Function called when listener detects new pos of mouse and sends the pos to client by protocol  
 :param x: x loc of mouse  
 :param y: y loc of mouse  
 """* self.server.send(self.clientIP, str(x).zfill(4) + str(y).zfill(4) + "4")  
  
 def \_on\_click(self, x: int, y: int, button, pressed):  
 *"""  
 Function checks if mouse clicked and sends gotten data to client by protocol  
 :param x: x pos of mouse click  
 :param y: y pos of mouse click  
 :param button: button to check mouse type click  
 :param pressed: check pressed or released  
 """* msg = str(x).zfill(4) + str(y).zfill(4)  
 if button == pynput.mouse.Button.left:  
 *# check if button was pressed or released and add to protocol accordingly* if pressed:  
 msg += "0"  
 else:  
 msg += "5"  
 elif button == pynput.mouse.Button.right:  
 *# check if button was pressed or released and add to protocol accordingly* if pressed:  
 msg += "1"  
 else:  
 msg += "6"  
  
 *# send data to client by protocol* self.server.send(self.clientIP, msg)  
  
 def \_on\_scroll(self, x: int, y: int, dx: int, dy: int):  
 *"""  
 Function checks and listens to mouse scroll wheel and sends to client data if mouse scrolled by protocol  
 :param x: x pos of mouse scroll  
 :param y: y pos of mouse scroll  
 :param dx: added x for mouse scroll  
 :param dy: added y for mouse scroll  
 """* msg = str(x).zfill(4) + str(y).zfill(4)  
 if dy > 0:  
 msg += "2"  
 else:  
 msg += "3"  
  
 *# send data to client by protocol* self.server.send(self.clientIP, msg)

# פרק ו' - UELP – בדיקות ('קופסא לבנה')

1. פירוט הבדיקות

* פירוט הבדיקות שהופיעו בפרק האפיון ברמת הביטים – לתאר בפירוט מה נדרש לבצע ומה בוצע בפועל. וכמובן על בדיקה צריכה להתבצע יותר מפעם אחת ועד שהיא עוברת

| **שם הבדיקה** | **מטרת הבדיקה** | **מה נדרש לבצע** | **מתי** | **מה בוצע בפועל** |
| --- | --- | --- | --- | --- |
| פתיחת תמיכה בין שני משתמשים | יצירת תמיכה חדשה עם קוד מהתומך | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך, בודקים שהקוד לא קרס והתמיכה החלה | 16.05.24 | המחשבים התחברו לסאשן והתמיכה החלה בלי בעיות |
| פתיחת תמיכה בין שני משתמשים | יצירת תמיכה חדשה עם קוד מהתומך | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך, בודקים שהקוד לא קרס והתמיכה החלה | 16.05.24 | המחשבים התחברו לסאשן והתמיכה החלה בלי בעיות |
| תמיכה פועלת עם מסך | בדיקה שרואים את מסך המבקש תמיכה בלי בעיות והוא משתנה בהתאם לשינוי המסך | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך, בודקים שרואים כהתחלה את מסך המבקש תמיכה אצל התומך. לאחר שרואים את המסך של המבקש תמיכה התומך יתחיל לבצע שינויים במסך שלו (פתיחת טאבים ואפליקציות כתיבת טקסט וכו) צריך לראות שבמסך של התומך רואים את השינויים מתקיימים. | 16.05.24 | המחשבים התחברו לסאשן, המבקש תמיכה ביצע שינויים במסך שלו והשינויים האלו נראו במסך של התומך בלי בעיות |
| תמיכה פועלת עם מסך | בדיקה שרואים את מסך המבקש תמיכה בלי בעיות והוא משתנה בהתאם לשינוי המסך | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך, בודקים שרואים כהתחלה את מסך המבקש תמיכה אצל התומך. לאחר שרואים את המסך של המבקש תמיכה התומך יתחיל לבצע שינויים במסך שלו (פתיחת טאבים ואפליקציות כתיבת טקסט וכו) צריך לראות שבמסך של התומך רואים את השינויים מתקיימים. | 16.05.24 | המחשבים התחברו לסאשן, המבקש תמיכה ביצע שינויים במסך שלו והשינויים האלו נראו במסך של התומך בלי בעיות |
| תמיכה פועלת עם עכבר | בדיקה של העכבר של התומך כשל המבקש תמיכה (כל פעולות העכבר) | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ומתחילים להזיז את העכבר של התומך. מסתכלים שהעכבר אצל המבקש תמיכה זז בהתאמה. לאחר מכן בודקים שבלחיצה ימנית ושמאלית בעכבר של התומך הלחיצה קוראת אצל המבקש תמיכה. ולבסוף נסובב את הscroll wheel של העכבר אצל התומך, נבדוק שהפעולה מתקיימת אצל המבקש תמיכה | 16.05.24 | המחשבים התחברו לסאשן, התומך ביצע פעולות בעכבר שלו והפעולות האלו יתקיימו אצל המבקש תמיכה בלי בעיות |
| תמיכה פועלת עם עכבר | בדיקה של העכבר של התומך כשל המבקש תמיכה (כל פעולות העכבר) | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ומתחילים להזיז את העכבר של התומך. מסתכלים שהעכבר אצל המבקש תמיכה זז בהתאמה. לאחר מכן בודקים שבלחיצה ימנית ושמאלית בעכבר של התומך הלחיצה קוראת אצל המבקש תמיכה. ולבסוף נסובב את הscroll wheel של העכבר אצל התומך, נבדוק שהפעולה מתקיימת אצל המבקש תמיכה | 16.05.24 | המחשבים התחברו לסאשן, התומך ביצע פעולות בעכבר שלו והפעולות האלו יתקיימו אצל המבקש תמיכה בלי בעיות |
| תמיכה פועלת עם מקלדת | בדיקה של המקלדת של התומך כשל המבקש תמיכה. (כל פעולות המקלדת) | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ומתחילים ללחוץ ולעזוב מקשים במקלדת של התומך. מסתכלים שהמקשים האלו נלחצים במחשב של המבקש תמיכה בהתאמה. לאחר מכן בודקים צירופים של כמה מקשים ביחד הידועים כמו למשל Ctrl + c. בודקים שהקומבינציה נלחצת ונעזבת במחשב של המבקש תמיכה כאשר היא נלחצת ונעזבת אצל התומך | 16.05.24 | המחשבים התחברו לסאשן, התומך ביצע פעולות על המקלדת שלו והפעולות האלו התקיימו במחשב של המבקש תמיכה בלי בעיות |
| תמיכה פועלת עם מקלדת | בדיקה של המקלדת של התומך כשל המבקש תמיכה. (כל פעולות המקלדת) | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ומתחילים ללחוץ ולעזוב מקשים במקלדת של התומך. מסתכלים שהמקשים האלו נלחצים במחשב של המבקש תמיכה בהתאמה. לאחר מכן בודקים צירופים של כמה מקשים ביחד הידועים כמו למשל Ctrl + c. בודקים שהקומבינציה נלחצת ונעזבת במחשב של המבקש תמיכה כאשר היא נלחצת ונעזבת אצל התומך | 16.05.24 | המחשבים התחברו לסאשן, התומך ביצע פעולות על המקלדת שלו והפעולות האלו התקיימו במחשב של המבקש תמיכה בלי בעיות |
| קיום תמיכה בין יותר מ2 מחשבים | בדיקה שהשרת מרובה לקוחות יכול לחבר יותר מ2 מחשבים לסאשנים שונים בלי בעיות | להתחבר למערכת על ידי ארבעה מחשבים שונים, בשניים מתחברים כתומך ובשניים האחרים כמבקש תמיכה, המבקשי תמיכה מכניסים את הקודים שמופיע במחשבים של התומכים (כל אחד אחד אחר) בו זמנית ומסתכלים שהמחשבים מצליחים להתחבר לסאשנים בלי בעיה וקריסות. | 16.05.24 | המחשבים התחברו כולם לסאשנים שלהם והתמיכות החלו בלי בעיה |
| קיום תמיכה בין יותר מ2 מחשבים | בדיקה שהשרת מרובה לקוחות יכול לחבר יותר מ2 מחשבים לסאשנים שונים בלי בעיות | להתחבר למערכת על ידי ארבעה מחשבים שונים, בשניים מתחברים כתומך ובשניים האחרים כמבקש תמיכה, המבקשי תמיכה מכניסים את הקודים שמופיע במחשבים של התומכים (כל אחד אחד אחר) בו זמנית ומסתכלים שהמחשבים מצליחים להתחבר לסאשנים בלי בעיה וקריסות. | 16.05.24 | המחשבים התחברו כולם לסאשנים שלהם והתמיכות החלו בלי בעיה |
| סיום תמיכה ע״י הנתמך | בדיקה שהמקלדת בצד המבקש תמיכה קולטת בקשת סיום סאשן ופועלת בהתאם | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ולוחצים על הקומבינציה Ctrl + Shift + d על המקלדת של המבקש תמיכה. מסתכלים במחשב של התומך והנתמך שפרוססים של תמיכה לא רצים יותר על המחשב ובמחשב של התומך בודקים שכבר אין לו שליטה על העכבר מקלדת ומסך של המבקש תמיכה. | 16.05.24 | המחשבים התחברו לסאשן, המבקש תמיכה לחץ על המקלדת שלו על הקומבינציה לסיום הסאשן, הפרוססים הופסקו אצל כל המחשבים והשליטה על המסך עכבר ומקלדת של המבקש תמיכה בצד התומך נעצרה גם היא בלי בעיה |
| סיום תמיכה ע״י הנתמך | בדיקה שהמקלדת בצד המבקש תמיכה קולטת בקשת סיום סאשן ופועלת בהתאם | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ולוחצים על הקומבינציה Ctrl + Shift + d על המקלדת של המבקש תמיכה. מסתכלים במחשב של התומך והנתמך שפרוססים של תמיכה לא רצים יותר על המחשב ובמחשב של התומך בודקים שכבר אין לו שליטה על העכבר מקלדת ומסך של המבקש תמיכה. | 16.05.24 | המחשבים התחברו לסאשן, המבקש תמיכה לחץ על המקלדת שלו על הקומבינציה לסיום הסאשן, הפרוססים הופסקו אצל כל המחשבים והשליטה על המסך עכבר ומקלדת של המבקש תמיכה בצד התומך נעצרה גם היא בלי בעיה |
| סיום תמיכה ע״י התומך | בדיקה שהמקלדת בצד התומך קולטת בקשת סיום סאשן ופועלת בהתאם | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ולוחצים על הקומבינציה Ctrl + Shift + d על המקלדת של התומך. מסתכלים במחשב של התומך והנתמך שפרוססים של תמיכה לא רצים יותר על המחשב ובמחשב של התומך בודקים שכבר אין לו שליטה על העכבר מקלדת ומסך של המבקש תמיכה. | 16.05.24 | המחשבים התחברו לסאשן, המבקש תמיכה לחץ על המקלדת שלו על הקומבינציה לסיום הסאשן, הפרוססים הופסקו אצל כל המחשבים והשליטה על המסך עכבר ומקלדת של המבקש תמיכה בצד התומך נעצרה גם היא בלי בעיה |
| סיום תמיכה ע״י התומך | בדיקה שהמקלדת בצד התומך קולטת בקשת סיום סאשן ופועלת בהתאם | להתחבר למערכת על ידי שני מחשבים שונים, באחד מתחברים כתומך ובשני כמבקש תמיכה, המבקש תמיכה מכניס את הקוד שמופיע במחשב של התומך. מתחברים לסאשן ולוחצים על הקומבינציה Ctrl + Shift + d על המקלדת של התומך. מסתכלים במחשב של התומך והנתמך שפרוססים של תמיכה לא רצים יותר על המחשב ובמחשב של התומך בודקים שכבר אין לו שליטה על העכבר מקלדת ומסך של המבקש תמיכה. | 16.05.24 | המחשבים התחברו לסאשן, המבקש תמיכה לחץ על המקלדת שלו על הקומבינציה לסיום הסאשן, הפרוססים הופסקו אצל כל המחשבים והשליטה על המסך עכבר ומקלדת של המבקש תמיכה בצד התומך נעצרה גם היא בלי בעיה |

# פרק ז' - UELP – מדריך למשתמש

1. מדריך למשתמש הכולל עבור כל תהליך/יכולת במערכת:

* הסבר על התהליך/יכולת
* תרשים זרימה (במקרה של תהליך)
* צילומי מסך מלווים בהסברים בשפה פשוטה

במידה ויש יותר ממשתמש אחד למערכת , לבנות פרקים שונים לדוגמא: מדריך למשתמש קצה, מדריך למנהל מערכת וכו'

מדריך להרצת השרת הראשי-

שלב 1- הורדת כל קבצי השרת

יש להוריד את כל קבצי השרת שצריך בשביל להריץ את השרת-

הקבצים:

* Database.py
* Sessioncodes.py
* Server\_logic.py
* Server\_protocol.py
* Servercomm.py
* Uelp\_db.sql
* AsymmetricEncryption.py
* SymmetricEncryption.py

**תמונה של קבצים אלו**

שלב 2- הרצת קובץ Server\_logic.py

יש להריץ את קובץ Server\_logic.py אשר יפעיל את השרת הראשי על המחשב שלכם (אפשר לדעת שהשרת רץ אם לא מופיעים שגיאות והשרת רץ בלי בעיות

מדריך להפעלת לקוח שמתחבר לשרת-

שלב 1- הורדת כל קבצי הלקוח

יש להוריד את כל קבצי הלקוח שצריך בשביל להריץ את הלקוח-

הקבצים:

* arrow\_off.png
* arrow\_on.png
* assistance\_seeker.png
* assistance\_seeker\_hover.png
* code.png
* connect.png
* connect\_off.png
* copied.png
* copy\_code.png
* copy\_code\_hover.png
* Helper\_choose.png
* Helper\_choose\_hover.png
* Info.png
* login.png
* login\_off.png
* logo.png
* pass.png
* signin.png
* signin\_off.png
* signup.png
* signup\_off.png
* user.png
* AssistanceSeeker\_keyboard\_logic.py
* AssistanceSeeker\_mouse\_logic.py
* AssistanceSeeker\_protocol.py
* AssistanceSeeker\_screen\_logic.py
* AsymmetricEncryption.py
* SymmetricEncryption.py
* Client\_logic.py
* Client\_protocol.py
* clientComm.py
* graphics.py
* Helper\_logic.py
* Helper\_protocol.py
* Helper\_screen\_logic.py
* Monitor\_Keyboard.py
* Monitor\_Mouse.py

שלב 2- שינוי כתובת הip לכתובת השרת

בשביל להריץ את הלקוח צריך להגדיר את כתובת ה ip שהלקוח מתחבר אליה ככתובת הip של המחשב שעליו רץ השרת שלכם.

נלך לקובץ Client\_logic.py ונשנה את הערך במשתנה ip בשורה 152 לכתובת הip של השרת שלנו.

שלב 3- הרצת קובץ Client\_logic.py

יש להריץ את קובץ Client\_logic.py לאחר ששינינו את כתובת הip בקובץ זה לכתובת של השרת שלנו.

נדע שהלקוח התחבר לשרת ורץ במידה ויפתח לנו חלון עם מסך התחברות שלמעלה שלו יהיה כתוב UELP

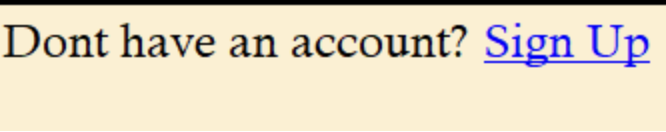
מדריך לפתיחת סאשן חדש וכניסה אליה-

בשביל ליצור סאשן שליטה ולהתחבר אליה נצטרך 2 מחשבים לפחות, מחשב אחד שיהיה התומך שלנו ומחשב נוסף שיהיה המבקש תמיכה שלנו . (מחשב התומך יכול גם להריץ את השרת או שאפשר להריץ את השרת על מחשב אחר) שני המחשבים האלו יצטרכו להתחבר למערכת כלקוחות.

שלב 1- מעבר למסך ההרשמה למערכת

בשביל להירשם למערכת נצטרך לעבור ממסך התחברות שפתוח לנו כרגע למסך ההרשמה.

יש לאתר את כפתור ה SignUp הכחול אשר נמצא מתחת לתיבות טקסט וללחוץ עליו בשביל לעבור למסך ההתחברות

****

שלב 2- הרשמה למערכת

בשביל להירשם למערכת יש להכניס שם משתמש וסיסמא

\*שימו לב הסיסמא שלכם חייבת להיות באורך של לפחות 8 תווים ולהכיל אות אחת גדולה באנגלית ותו מיוחד\*

לאחר שנכניס שם משתמש וסיסמא נלחץ על כפתור ה Sign Up הגדול שצריך להיות דלוק בצבע שחור לאחר שהכנסתם שם משתמש וסיסמא.

במידה והשם משתמש וסיסמא שלכם מאושרים תעברו למסך חדש אם לא, תקבלו הודעה שתסביר לכם מה אתם צריכים לשנות.

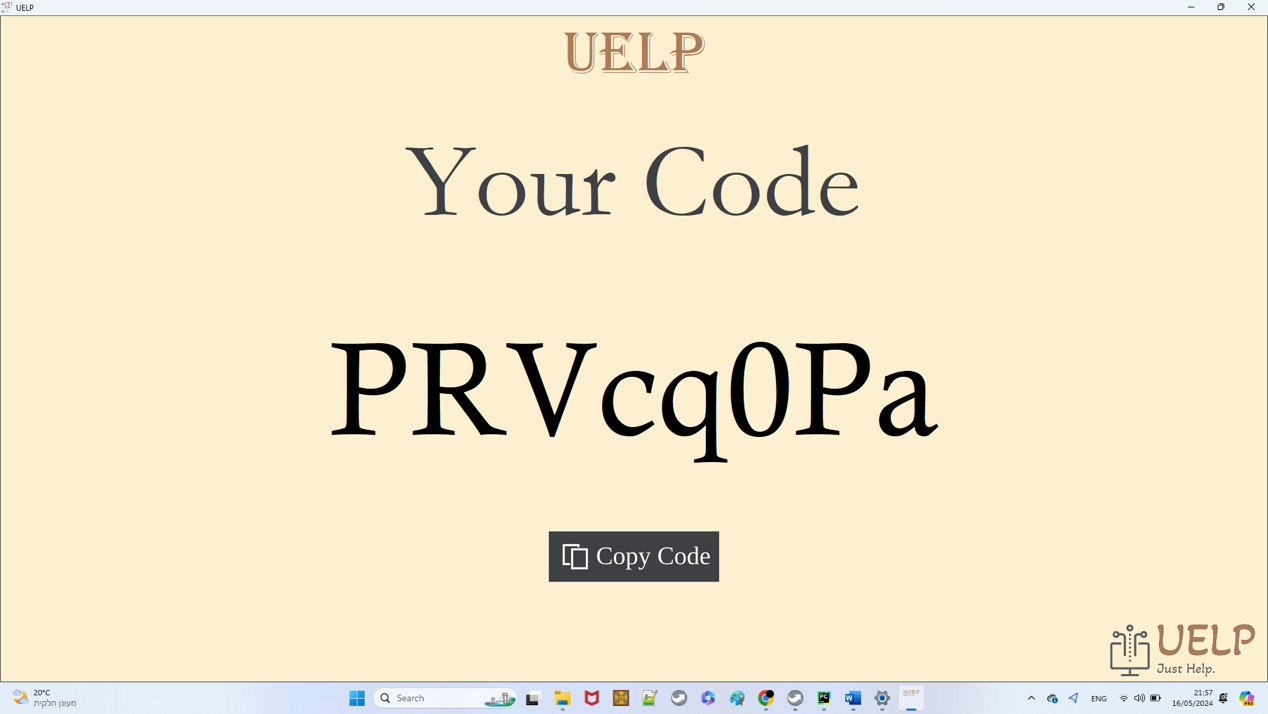


שלב 3- בחירת סוג משתמש

בשלב זה נתפצל וכל אחד מהמחשבים יעשה משהו שונה. המחשב שמבחינתכם אתם רוצים שיהיה התומך צריך ללחוץ על הכפתור השמאלי הגדול שרשום עליו Helper. המחשב שהוא המבקש תמיכה צריך ללחוץ על כפתור הימני שרשום עליו Assistance Seeker.

****

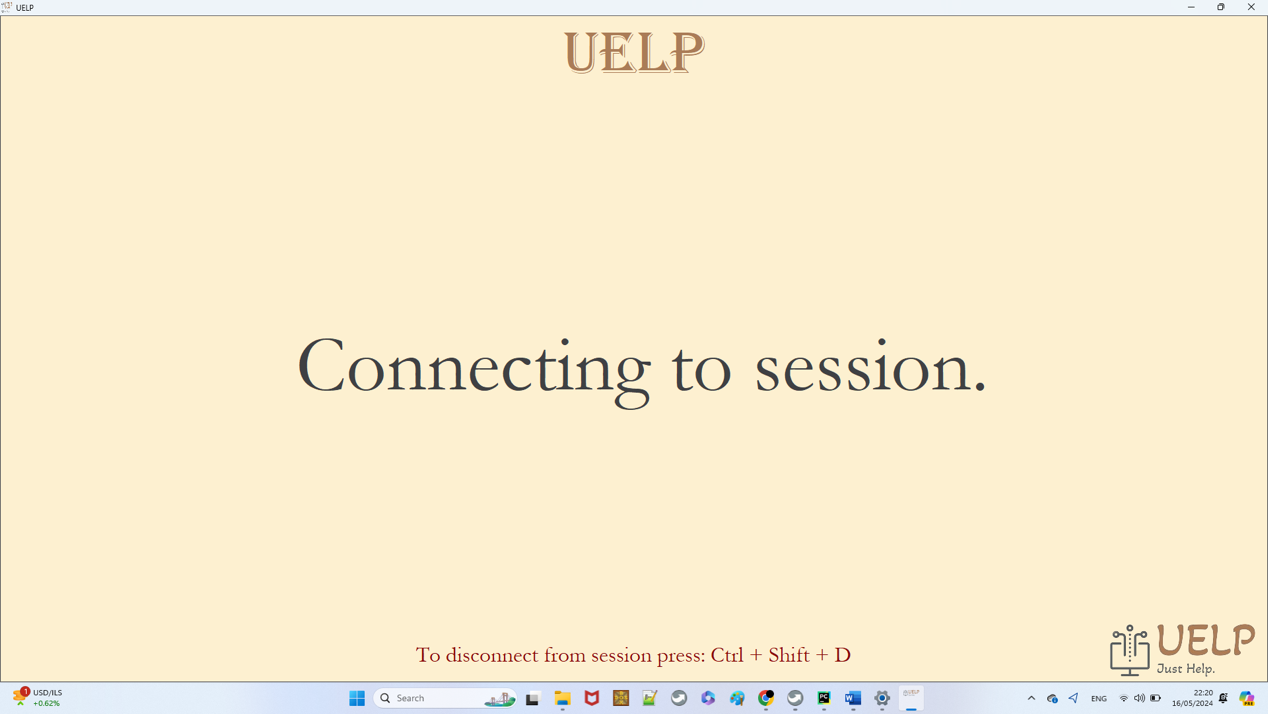
שלב 4- הכנסת קוד

בשלב הזה על מסך של המחשב התומך נראה קוד באורך 8 תווים ועל מסך המבקש תמיכה נראה תיבת טקסט וכפתור ״קונקט״ גדול. נכניס את הקוד אשר כתוב על המסך של תומך לתיבת הטקסט ונלחץ על הכפתור.



כעת במידה והכנסתם את הקוד נכון תועברו בשני המחשבים למסך התחברות. במסך זה אנחנו טוענים להתחבר לסאשן ויהיה כתוב לנו למטה איך אנחנו מתנתקים מהסאשן שליטה במידה ואנחנו רוצים.

נתנתק מהסאשן בעזרת הקומבינציה בלחיצה ביחד על המקלדת, גם כתומך וגם כמבקש תמיכה.

Control + shift + d

לאחר מספר שניות המסך יעלם, אצל התומך נוכל לראות שאנחנו רואים את המסך של המבקש תמיכה ואנחנו יכולים להזיז לו את העכבר וללחוץ על המקלדת ועכבר שלו ואצל המבקש תמיכה נוכל לראות שהפעולות האלו מתקיימות ולעבוד ביחד עם התומך על המחשב שלנו.

העבודה שלי על הפרויקט השנה הייתה מאתגרת מעניינת ופשוט כיפית מכל מובן. השנה לפרויקט הכי גדול בכל השלוש שנים במגמה ידעתי שאני רוצה לבחור פרויקט שיאתגר אותי ואני ייהנה ממנו בו זמנית. בשנים קודמות הרגשתי שמאוד השקעתי בפרויקטים שלי אך בגלל הזמן הקצר שעבדנו עליהם לא נכנסתי לעומק כמו שנכנסתי עם הפרויקט הזה. לכן כשמרי הציע לי את הרעיון של מערכת לשליטה ובקרה על מחשבים מרחוק ישר ידעתי שזה יכול להיות משהו שיתאים לתנאים שהצבתי לעצמי.

אני זוכר שבתחילת העדוה על הפרויקט פחדתי מאוד מהאתגרים שעמדו בפני שכתבתי אותם בתיק פרויקט, בגלל גודל הפרויקט פחדתי שלא אצליח להתמודד עם הכל ביחד ולא ידעתי אם אצליח לסיים ביעד. אם עבודה חכמה ויעילה הצלחתי לעבור כל אחד מהפחדים האלו. אני מאוד מאוד מאוד השקעתי בתיק פרויקט כי מרי המליצה לנו להגיע הכי מוכנים שאפשר לקוד והעמדתי את זה כמטרה בשבילי. אני יודע שעשיתי בדיוק את זה כי הפחדים שהיו לי מבחינת הקוד כמו שליטה על מקלדת מסך ועכבר עברו תוך כמה שבועות מכיוון שהגעתי כל כך מוכן אליהם. ידעתי מה אני רוצה לבנות ואיך אני הולך לעשות את זה ועכשיו רק נשאר להתאים את הדבר הזה לתוך קוד.

אירוע אחד מאוד מעניין שקרא השנה היה הגרפיקה של הפרויקט שלי. בתחילת הדרך ידעתי שאני רוצה לאתגר את עצמי וליצור פרויקט שיש לי הכי פחות ידע בו ולהמשיך לאתגר את עצמי לאורך השנה. ידעתי שגרפיקה זה משהו יותר צדדי בשבילי, אך ככל שעבדתי יותר על הפרויקט והגעתי לגרפיקה הבנתי כמה זה חשוב לי וכמה זה טוב שיש לך מערכת יפה ואסתטית. הגעתי ליעדים שלי מבחינת הפרויקט עצמו יחסית מוקדם מכיוון שעבדתי קשה לכן זה נתן לי המון זמן לעבוד על הגרפיקה שלי וליצור תוצר יפה שאני מאוד שמח ומרוצה ממנו.

עוד אירוע מעניין שקרה בשנה הוא הפיתוח והשילוב שלנו כמגמה. לפני העבודה על הפרויקטים הייתה אווירה טובה בשיעורים אך לא הרגשתי שאנחנו תורמים אחד לשני באופן קיצוני. השנה במהלך העבודה על הפרויקט הרגשתי שכולם עזרו לכולם, כלומר אני וחברי למגמה היינו מפצחים ביחד בעיות ובאגים אחד של השני והרגיש כאילו כולם מעודכנים על ההתקדמות של כולם שהביא הרבה מוטיבציה לעבוד קשה ולהתקדם ביחד.

אני רוצה להודות תודה ראשונה למרי גבע, המורה שלי לסייבר שליוותה אותי לאורך כל השנתיים האחרונות במגמה ותמכה בי ועזרה לי שעות רבות. תודה רבה לך מרי אני מעריך אותך מאוד ולא הייתי מצליח להגיע לתוצר הסופי שהגעתי אליו בלעדייך.

שנית אני רוצה להודות לחברי למגמה שעזרו לי לא פחות ושהצלחנו ליצור ביחד אווירה טובה בשיעורים ולקדם אחד את השני.

אני רוצה גם להודות למשפחה שלי שתמכה בי לאורך כל הדרך.

העבודה השנה על הפרויקט פיתחה אותי ועזרה לי לראות את היכולות שלי במקצוע זה בצורה אחרת, אני כל כך שמח שעבדתי קשה ושנתתי מעצמי לגמרי למען המטרה הזו.

# פרק ט' - UELP – ביבליוגרפיה

1. רקע תיאורטי
2. ספרות מקצועית ספציפית לנושא העבודה (רשימת ספרים, מאמרים והפנייה לכתובות אתרים המכילים חומר רקע ששימש לפיתוח העבודה)
3. הערה: לא מספיק להתייחס לספרים המיועדים ללימוד שפה או כלים יישומיים ומדריכים למיניהם.
4. קישורים לאתרים באינטנרט – קישורים ל stackOverflow עם הסבר על הנושא שנחקר (לא תחביר...)

# הקוד המלא-

Encryptions:

import base64  
import hashlib  
  
from Cryptodome import Random  
from Cryptodome.Cipher import AES  
from Cryptodome.Util.Padding import pad, unpad  
  
  
*# class to work with AES encryption and hash*class AES\_hash\_cipher:  
 def \_\_init\_\_(self, key: str):  
 *"""  
 builder function to create a new object of AES\_hash\_cipher with a key  
 :param key: key to use for both sides  
 """* self.bs = AES.block\_size  
 self.key = hashlib.sha256(key.encode()).digest()  
  
 def encrypt(self, raw: bytes) -> bytes:  
 *"""  
 method encrypts raw data gotten with AES encryption of key  
 :param raw: data to encrypt  
 :return: encoded encrypted data  
 """* raw = pad(raw, AES.block\_size)  
 iv = Random.new().read(AES.block\_size)  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
  
 return base64.b64encode(iv + cipher.encrypt(raw))  
  
 def decrypt(self, enc: bytes) -> str:  
 *"""  
 method decrypts encrypted data gotten with AES decryption of key  
 :param enc: encoded encrypted data  
 :return: encoded decrypted data  
 """* enc = base64.b64decode(enc)  
 iv = enc[:AES.block\_size]  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
  
 return unpad(cipher.decrypt(enc[AES.block\_size:]), AES.block\_size).decode('utf-8')  
  
 @staticmethod  
 def hash(data: str) -> bytes:  
 *"""  
 method gets data and uses SHA-256 to return it's hash  
 :param data: data to use hash function on  
 :return: encoded hashed data  
 """* return hashlib.sha256(data.encode()).digest()

from cryptography.hazmat.backends import default\_backend  
from cryptography.hazmat.primitives import serialization, hashes  
from cryptography.hazmat.primitives.asymmetric import rsa, padding  
  
  
*# class to work with Asymmetric RSA encryption*class RSA\_cipher:  
  
 def \_\_init\_\_(self):  
 *"""  
 builder function to create a new object of RSA\_cipher with RSA private and public keys  
 """* self.privateKey = rsa.generate\_private\_key(public\_exponent=65537, key\_size=2048, backend=default\_backend())  
 self.publicKey = self.privateKey.public\_key()  
  
 @staticmethod  
 def encrypt(data: str, public\_key\_str: str) -> bytes:  
 *"""  
 method to encrypt data with a RSA public key  
 :param data: data to encrypt  
 :param public\_key\_str: String representation of an RSA public key  
 :return: encrypted data in bytes  
 """* try:  
 *# Convert string representation to RSA public key* public\_key = serialization.load\_pem\_public\_key(public\_key\_str.encode(), backend=default\_backend())  
  
 *# Encrypt the message with the public key* enc = public\_key.encrypt(data.encode(), padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),  
 algorithm=hashes.SHA256(), label=None))  
 except Exception as e:  
 print(e)  
 print("The given String key does not match the RSA public key format")  
 else:  
 return enc  
  
 def decrypt(self, enc\_data: bytes) -> str:  
 *"""  
 method to decrypt, encrypted data gotten using the private key of the class  
 :param enc\_data: data encrypted using the public key  
 :return: decrypted data as string  
 """* try:  
 *# Decrypt the message with the private key* decrypted\_data = self.privateKey.decrypt(enc\_data,  
 padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()),  
 algorithm=hashes.SHA256(), label=None))  
 except Exception as e:  
 print(e)  
 print("The given encrypted data does not match RSA key size")  
 else:  
 return decrypted\_data.decode()  
  
 def get\_string\_key(self):  
 *"""  
 method gets a string representation of the public key  
 :return: String representation of the public key  
 """  
 # Get the string representation of the public key* stringkey = self.publicKey.public\_bytes(encoding=serialization.Encoding.PEM,  
 format=serialization.PublicFormat.SubjectPublicKeyInfo).decode()  
  
 return stringkey

Server code:

def unpackData(data: str) -> str and list:  
 *"""  
 function gets data and unpacks it based on known opcodes and params protocols  
 :param data: data to unpack  
 :return: the opcode of the data and a list of all it's params  
 """* opcode = data[:2]  
 params = []  
  
 if opcode == "00" or opcode == "01":  
 username\_len = int(data[2])  
 params.append(data[3:3 + username\_len])  
 params.append(data[3 + username\_len:])  
 elif opcode == "02" or opcode == "04" or opcode == "06":  
 params.append(data[2:])  
  
 return opcode, params  
  
  
def pack\_login\_ans(authorized: str) -> str:  
 *"""  
 function packs login ans based on the protocol  
 :param authorized: True if login was authorized False otherwise  
 :return: packed data based on protocol  
 """* return "00" + authorized  
  
  
def pack\_signup\_ans(authorized: bool) -> str:  
 *"""  
 function packs signup ans based on the protocol  
 :param authorized: True if signup was authorized False otherwise  
 :return: packed data based on protocol  
 """* return "010" if authorized else "011"  
  
  
def pack\_typeuser\_ans(authorized: bool) -> str:  
 *"""  
 function packs type user ans based on the protocol  
 :param authorized: True if type user was authorized False otherwise  
 :return: packed data based on protocol  
 """* return "020" if authorized else "021"  
  
  
def pack\_getcode\_ans(code: str) -> str:  
 *"""  
 function packs get code request ans based on the protocol  
 :param code: code to send  
 :return: packed data based on protocol  
 """* return "03" + code  
  
  
def pack\_expired\_code() -> str:  
 *"""  
 function packs expired code msg based on the protocol  
 :return: packed data based on protocol  
 """* return "032"  
  
  
def pack\_code\_ans(authorized: bool) -> str:  
 *"""  
 function packs code given ans based on the protocol  
 :param authorized: True if code gotten was authorized False otherwise  
 :return: packed data based on protocol  
 """* return "040" if authorized else "041"  
  
  
def pack\_con\_data(ip: str, userType: str) -> str:  
 *"""  
 function packs connection data msg based on the protocol  
 :param ip: ip to send  
 :param userType: user type of client sending connection data too  
 :return: packed data based on protocol  
 """* return "05" + str(len(ip)).zfill(2) + ip + userType

import select  
import socket  
import threading  
  
import Helper\_protocol  
from AsymmetricEncryption import RSA\_cipher  
from SymmetricEncryption import AES\_hash\_cipher  
  
  
*# class to communicate and work with clients of multiclient server*class Server\_comm:  
 def \_\_init\_\_(self, recv\_q, port: int, bindIP: str = None):  
 *"""  
 Builder method creates a new "Server\_comm" object with a Queue, port and bindIP  
 :param recv\_q: Queue for messages to the logic  
 :param port: port that server will run on  
 :param bindIP: IP that server will check if it uses port 2001, 2002, 2003  
 """* self.recv\_q = recv\_q  
 self.port = port  
 self.socket = socket.socket()  
 self.bindIP = bindIP  
 self.open\_clients = {}  
 self.is\_running = False  
 self.RSAobject = RSA\_cipher()  
 threading.Thread(target=self.\_main\_loop).start()  
  
 def \_main\_loop(self):  
 *"""  
 Function runs server main loop, connects clients creates shared keys with them and gets msgs from them  
 and encrypts them for logic  
 """* self.socket.bind(("0.0.0.0", self.port))  
 self.socket.listen(3)  
 self.is\_running = True  
  
 while self.is\_running:  
 rlist, wlist, xlist = select.select([self.socket] + list(self.open\_clients.keys()),  
 list(self.open\_clients.keys()), [], 0.03)  
  
 for current\_socket in rlist:  
 *# new client* if current\_socket is self.socket:  
 client, addr = self.socket.accept()  
 print(f"{addr[0]} - connected")  
 *# check if server is keyboard, mouse or screen server, if yes check if ip connected is bindIP* if self.port != 2000 and self.bindIP != addr[0]:  
 self.\_disconnect\_client(client)  
  
 *# create a new shared key with client using RSA and AES encryption* threading.Thread(target=self.\_get\_shared\_key, args=(addr[0], client)).start()  
 else:  
 *# get data len of client data and client data and decrypt* try:  
 datalen = int(current\_socket.recv(3).decode())  
 data = current\_socket.recv(int(datalen))  
 except Exception as e:  
 print(str(e))  
 print("main server in server conn")  
 self.\_disconnect\_client(current\_socket)  
 continue  
  
 data = self.open\_clients[current\_socket][1].decrypt(data)  
 *# if data is from screen port call recvImage otherwise put in recv\_q for logic* if self.port == 2003:  
 self.\_recvImage(current\_socket, data)  
 else:  
 self.recv\_q.put((data, self.open\_clients[current\_socket][0]))  
  
 def \_get\_shared\_key(self, clientIP: str, curSocket):  
 *"""  
 Function sends servers public key to new client and gets and saves the shared key of server and client  
 :param clientIP: ip of client to create shared key with  
 :param curSocket: socket of client to create shared key with  
 """  
 # send servers public key to client and get a shared encrypted key from client* try:  
 curSocket.send(self.RSAobject.get\_string\_key().encode())  
 sharedKey = curSocket.recv(256)  
 except Exception as e:  
 print(e)  
 print("in get shared key, server comm")  
 self.\_disconnect\_client(curSocket)  
 else:  
 *# check length of key is key gotten and decrypt the shared key with private key and save it* if len(sharedKey) != 256:  
 self.\_disconnect\_client(curSocket)  
 else:  
 sharedKey = self.RSAobject.decrypt(sharedKey)  
 self.open\_clients[curSocket] = (clientIP, AES\_hash\_cipher(sharedKey))  
  
 def send(self, ip: str, msg: str):  
 *"""  
 Send encrypted data to certain client  
 :param ip: ip of client to send msg too  
 :param msg: msg to send to client  
 """* if self.is\_running:  
 client = self.\_find\_socket\_by\_ip(ip)  
 if client is not None:  
 *# if client exists in open clients encrypt with AES and send with length by protocol* if client in self.open\_clients.keys():  
  
 msg = self.open\_clients[client][1].encrypt(msg.encode())  
 try:  
 client.send(str(len(msg)).zfill(3).encode() + msg)  
 except Exception as e:  
 print(str(e))  
 self.\_disconnect\_client(client)  
  
 def \_disconnect\_client(self, client):  
 *"""  
 Function disconnects client from server, by closing socket and removing from dictionarys  
 :param client: socket of client to disconnect  
 """  
 # if client in open\_clients remove him and tell logic by sending disconnect* if client in self.open\_clients.keys() and self.recv\_q:  
 print(f"{self.open\_clients[client]} - disconnect")  
 self.recv\_q.put(("disconnect", self.open\_clients[client][0]))  
 *# if client disconnected and the server is keyboard mouse or screen helper server (based on port)  
 # close server as session is closed because Assistance seeker closed client* if (self.port == 2001 or self.port == 2002 or self.port == 2003) and self.open\_clients[client][0] \  
 == self.bindIP:  
 self.close\_server()  
 del self.open\_clients[client]  
 client.close()  
  
 def \_find\_socket\_by\_ip(self, findip: str):  
 *"""  
 function find clients socket by ip  
 :param findip: ip to find it's socket  
 :return: socket of clients ip  
 """* client = None  
 for soc, ip\_enc in self.open\_clients.items():  
 if findip == ip\_enc[0]:  
 client = soc  
 break  
 return client  
  
 def close\_server(self):  
 *"""  
 End main loop  
 """* self.is\_running = False  
  
 def is\_running(self):  
 *"""  
 Check if server is running  
 :return: True or False if server is running  
 """* return self.is\_running  
  
 def closeClient(self, clientIP: str):  
 *"""  
 Function closes a client  
 :param clientIP: ip of client to close  
 """* client = self.\_find\_socket\_by\_ip(clientIP)  
  
 if client is not None:  
 self.\_disconnect\_client(self.\_find\_socket\_by\_ip(clientIP))  
  
 def \_recvImage(self, client, data: str):  
 *"""  
 Function is called when port is screen port and data was gotten from client of bindIP meaning  
 this is a screen server and an Image is being recieved after data gotten  
 :param client: client gotten data from  
 :param data: data gotten from client for screen  
 """* if client and client in self.open\_clients.keys():  
 client\_ip = self.open\_clients[client][0]  
 *# unpack data based on protocol* opcode, params = Helper\_protocol.unpackData(data)  
 file\_is\_ok = True  
 image\_data\_size = ""  
 *# if opcode is 01 we got a part image, if it is 02 we got a full Image  
 # take out data accordingly* if opcode == "01":  
 if len(params) == 5:  
 top = params[0]  
 left = params[1]  
 bottom = params[2]  
 right = params[3]  
 image\_data\_size = params[4]  
 else:  
 self.\_disconnect\_client(client)  
 file\_is\_ok = False  
 elif opcode == "02":  
 if len(params) == 1:  
 image\_data\_size = params[0]  
 else:  
 self.\_disconnect\_client(client)  
 file\_is\_ok = False  
 else:  
 self.\_disconnect\_client(client)  
 file\_is\_ok = False  
  
 *# check image data size gotten* if not image\_data\_size.isnumeric():  
 self.\_disconnect\_client(client)  
 file\_is\_ok = False  
  
 if file\_is\_ok:  
 image\_data\_size = int(image\_data\_size)  
  
 *# receive file data* file = bytearray()  
 while len(file) < image\_data\_size and file\_is\_ok:  
 size = image\_data\_size - len(file)  
 if size > 1024:  
 try:  
 file.extend(client.recv(1024))  
 except Exception as e:  
 self.\_disconnect\_client(client)  
 file\_is\_ok = False  
 else:  
 try:  
 file.extend(client.recv(size))  
 except Exception as e:  
 self.\_disconnect\_client(client)  
 file\_is\_ok = False  
 break  
  
 *# put image header data and image data in queue* if file\_is\_ok:  
 all\_image\_data = []  
 if len(params) == 5:  
 all\_image\_data.append(top)  
 all\_image\_data.append(left)  
 all\_image\_data.append(bottom)  
 all\_image\_data.append(right)  
 all\_image\_data.append(file)  
 self.recv\_q.put((all\_image\_data, client\_ip))

import base64  
import secrets  
import threading  
import time  
  
  
*# class to work with session codes of sessions valid in server*class Session\_codes:  
 def \_\_init\_\_(self, expired\_q):  
 self.expired\_q = expired\_q  
 self.codes = {}  
 self.cleanseTime = 180  
 threading.Thread(target=self.\_remove\_expired).start()  
  
 def \_remove\_expired(self):  
 *"""  
 Function checks dictionary of codes every two seconds, if there are codes who expired remove from dic and add to  
 expired codes queue  
 """* while True:  
 currTime = time.time()  
 expired\_ips = []  
  
 for ip in self.codes.keys():  
 createTime = self.codes[ip][1]  
 if currTime - createTime > self.cleanseTime:  
 expired\_ips.append(ip)  
 self.expired\_q.put(ip)  
  
 *# remove expired\_ips from dictionary* for ip in expired\_ips:  
 del self.codes[ip]  
  
 time.sleep(2)  
  
 def checkCode(self, code: str) -> bool:  
 *"""  
 Function checks if code gotten exists in dictionary of all codes  
 :param code: code to check if in dic  
 :return: True if code in codes dictionary, False otherwise  
 """* onlyCodes = [codeInfo[0] for codeInfo in self.codes.values()]  
 return code in onlyCodes  
  
 def createCode(self, ip: str) -> str:  
 *"""  
 Function creates a random string code that dosent exist in all codes and adds it to dic  
 :param ip: ip of user whose code was created for (saved in dic with code)  
 :return: code created  
 """  
 # getting random code and checking if exists* while True:  
 newCode = base64.b64encode(secrets.token\_bytes(6)).decode()  
 if not self.checkCode(newCode):  
 break  
  
 self.codes[ip] = (newCode, time.time())  
 return newCode  
  
 def ip\_from\_code(self, code: str) -> str:  
 *"""  
 Function returns the ip of a user from a code gotten  
 :param code: code of session that a user created  
 :return: ip of user that created the session code  
 """* ip = None  
 for ip\_dic, code\_time in self.codes.items():  
 if code == code\_time[0]:  
 ip = ip\_dic  
 break  
 return ip

import sqlite3  
  
from SymmetricEncryption import AES\_hash\_cipher  
  
  
*# class to work with Database and tables of Uelp\_db.sql*class Database\_comm:  
 def \_\_init\_\_(self):  
 *"""  
 builder function creates a new Database\_comm object with table and database names  
 """* self.DBname = "Uelp\_db.sql"  
 self.conn = None  
 self.curr = None  
 self.loginTableName = "USERS"  
 self.blacklistTableName = "MACS"  
 self.\_create\_db()  
  
 def \_create\_db(self):  
 *"""  
 function creates new tables USERS and MACS if thy don't already exist  
 """  
  
 # connect to DB* self.conn = sqlite3.connect(self.DBname)  
 self.curr = self.conn.cursor()  
 *# create tables* self.curr.execute(  
 "CREATE TABLE IF NOT EXISTS " + self.loginTableName + "(username VARCHAR(25), password VARCHAR(32),"  
 " PRIMARY KEY(username))")  
 self.curr.execute(  
 "CREATE TABLE IF NOT EXISTS " + self.blacklistTableName + "(mac VARCHAR(32), PRIMARY KEY(mac))")  
  
 self.conn.commit()  
  
 def \_checkUser(self, username: str) -> bool:  
 *"""  
 function checks if username exists in USERS table  
 :param username: username gotten from user  
 :return: True if username exists in USERS and False otherwise  
 """* self.curr.execute("SELECT username FROM " + self.loginTableName + " WHERE username = ?", (username,))  
 return self.curr.fetchone() is not None  
  
 def addUser(self, username: str, password: str) -> bool:  
 *"""  
 function adds username and hashed password to USERS table if the username does not already exist  
 :param username: username to add to USERS  
 :param password: password to add to USERS  
 :return: True if successfully added and False otherwise  
 """  
  
 # check if username already exists in table if it does not add it and his hashed password into USERS* if len(username) < 26 and not self.\_checkUser(username):  
 self.curr.execute("INSERT INTO " + self.loginTableName + " (username, password) VALUES (?, ?)",  
 (username, AES\_hash\_cipher.hash(password)))  
 self.conn.commit()  
 ans = True  
 else:  
 ans = False  
 return ans  
  
 def checkPassword(self, username: str, password: str) -> bool:  
 *"""  
 function checks if usernames password gotten exists in table  
 :param username: username to check in USERS  
 :param password: password to check in USERS  
 :return: True if password of username gotten exists for username in USERS  
 and False if username or password don't exist in USERS  
 """* if len(username) < 26 and self.\_checkUser(username):  
 self.curr.execute("SELECT password FROM " + self.loginTableName + " WHERE username = ? and password = ?",  
 (username, AES\_hash\_cipher.hash(password)))  
 ans = self.curr.fetchone() is not None  
 else:  
 ans = False  
 return ans  
  
 def macExists(self, mac: str) -> bool:  
 *"""  
 function checks if mac gotten exists in MACS table  
 :param mac: mac to check if it's hash exists in MACS  
 :return: True if exists False otherwise  
 """* self.curr.execute("SELECT mac FROM " + self.blacklistTableName + " WHERE mac = ?", (AES\_hash\_cipher.hash(mac),))  
 return self.curr.fetchone() is not None  
  
 def addBlackMac(self, mac: str) -> bool:  
 *"""  
 function checks if mac gotten exists in MACS and if dosen't adds it's hash  
 :param mac: mac to add to MACS table  
 :return: True if mac was added successfully added and False otherwise  
 """* if not self.macExists(mac):  
 self.curr.execute("INSERT INTO " + self.blacklistTableName + " (mac) VALUES (?)",  
 (AES\_hash\_cipher.hash(mac),))  
 self.conn.commit()  
 ans = True  
 else:  
 ans = False  
 return ans  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 db = Database\_comm()  
 print(db.addBlackMac("64:00:6A:42:93:94"))

import queue  
import threading  
  
from scapy.layers.l2 import getmacbyip  
  
import Server\_protocol as Protocol  
from Database import Database\_comm  
from Servercomm import Server\_comm  
from Sessioncodes import Session\_codes  
  
  
def handleMsgs(server, recv\_q):  
 *"""  
 Function creates database comm object and recieves new msgs from server queue unpacks the msg based on protocol  
 and calls function by opcode  
 :param server: server object to communicate with server  
 :param recv\_q: servers msgs queue  
 """  
 # create database object to communicate with database* db = Database\_comm()  
  
 while True:  
 data, ip = recv\_q.get()  
 print(data)  
 *# if data is disconnect call disconnect client function* if data == "disconnect":  
 disconnectClient(ip, server)  
 else:  
 opcode, params = Protocol.unpackData(data)  
 *# call function based on opcode gotten in data and command dic* commands[opcode](ip, db, server, params)  
  
  
def handleExpired(server, expired\_q):  
 *"""  
 Function checks expired codes queue and gets expired code sessions ips and sends expired code protocol to the  
 ip goten  
 :param server: server object to communicate with server  
 :param expired\_q: queue that has ips of users with expired session codes  
 """* while True:  
 ip = expired\_q.get()  
 server.send(ip, Protocol.pack\_expired\_code())  
  
  
def handle\_mac\_addr(clientIP, db, server, params):  
 *"""  
 Function adds mac address of new user to users dictionary with ip  
 :param clientIP: ip of users mac addr  
 :param db: database object to communicate with database  
 :param server: server object to communicate with server  
 :param params: list of paramaters gotten with opcode from user  
 """* mac\_addr = params[0]  
  
 users[clientIP] = [mac\_addr, None, None]  
  
  
def handle\_login(clientIP, db, server, params):  
 *"""  
 Function gets username and password of user checks his password by database and sends msg accordingly back to user  
 based on if username and password fit credentials  
 :param clientIP: ip of user from whom gotten login msg  
 :param db: database object to communicate with database  
 :param server: server object to communicate with server  
 :param params: list of paramaters gotten with opcode from user  
 """* username = params[0]  
 password = params[1]  
  
 if clientIP in users.keys():  
 *# check if password of username gotten exists in db* userOK = db.checkPassword(username, password)  
 if userOK:  
 *# check if user is already logged in and set opcode accordingly* if check\_logged\_in(username):  
 userOK = "2"  
 else:  
 *# add username to users dic and set opcode if user can log in successfully* users[clientIP][2] = username  
 userOK = "0"  
 else:  
 userOK = "1"  
 else:  
 userOK = "1"  
  
 *# send msg back to user based on if user can login or not* server.send(clientIP, Protocol.pack\_login\_ans(userOK))  
  
  
def handle\_signup(clientIP, db, server, params):  
 *"""  
 Function gets username and password of user checks his password by database and sends msg accordingly back to user  
 based on if username and password fit credentials  
 :param clientIP: ip of user from whom gotten signup msg  
 :param db: database object to communicate with database  
 :param server: server object to communicate with server  
 :param params: list of paramaters gotten with opcode from user  
 """* username = params[0]  
 password = params[1]  
  
 if clientIP in users.keys():  
 *# check if can add the user with the username and password to db successfully* newuserOK = db.addUser(username, password)  
 if newuserOK:  
 *# add username to users dic if created and added to db table* users[clientIP][2] = username  
 else:  
 newuserOK = False  
  
 *# send msg back to user based on if user can signup or not* server.send(clientIP, Protocol.pack\_signup\_ans(newuserOK))  
  
  
def check\_logged\_in(username):  
 *"""  
 Function checks if user is logged in by checking users dic and seeing if users username exists in this dic  
 :param username: username of user to check if logged in  
 :return: True if user is logged in and false if not  
 """* for userData in users.values():  
 if username == userData[2]:  
 ans = True  
 break  
 else:  
 ans = False  
 return ans  
  
  
def handle\_typeUser(clientIP, db, server, params):  
 *"""  
 Function gets user type asked from user and checks if users mac and mac from arp request is the same and ok  
 accordingly the function checks if the user type asked fits the database blacklist credentials  
 and returns msg to user accordingly  
 :param clientIP: ip of user whom gotten typeuser msg from  
 :param db: database object to communicate with database  
 :param server: server object to communicate with server  
 :param params: list of paramaters gotten with opcode from user  
 """* typeUser = params[0]  
 *# mac gotten from sending arp request to users ip* mac\_from\_arp = getmacbyip(clientIP)  
  
 *# check if mac from arp equals mac gotten from user or your own computer* if mac\_from\_arp == "ff:ff:ff:ff:ff:ff" or mac\_from\_arp == users[clientIP][0]:  
 *# if user wants to be a helper check if users mac is in blacklist based on db table  
 # save user type and set msg to user accordingly* if typeUser == "0":  
 if not db.macExists(users[clientIP][0].upper()):  
 users[clientIP][1] = "H"  
 typeUser\_ans = True  
 else:  
 typeUser\_ans = False  
 elif typeUser == "1":  
 users[clientIP][1] = "A"  
 typeUser\_ans = True  
 msg = Protocol.pack\_typeuser\_ans(typeUser\_ans)  
 else:  
 msg = Protocol.pack\_typeuser\_ans(False)  
  
 *# send msg back to user by protocol of if user type is verified or not* server.send(clientIP, msg)  
  
  
def handle\_getCode(clientIP, db, server, params):  
 *"""  
 Function checks if user is Helper and sends msg back to user with protocol and a new session code  
 :param clientIP: ip of user whom gotten getcode msg from  
 :param db: database object to communicate with database  
 :param server: server object to communicate with server  
 :param params: list of paramaters gotten with opcode from user  
 """* if users[clientIP][1] == "H":  
 msg = Protocol.pack\_getcode\_ans(codes.createCode(clientIP))  
 server.send(clientIP, msg)  
  
  
def handle\_codeCheck(clientIP, db, server, params):  
 *"""  
 Function gets code user entered from params and checks if code exists and returns msg to user accordingly  
 :param clientIP: ip of user whom gotten check code msg from  
 :param db: database object to communicate with database  
 :param server: server object to communicate with server  
 :param params: list of paramaters gotten with opcode from user  
 """* userCode = params[0]  
  
 *# check if user code gotten from is Assistance Seeker* if users[clientIP][1] == "A":  
 *# check if code gotten exists as a valid session code and send msg back to user accordingly* codeAnswer = codes.checkCode(userCode)  
 server.send(clientIP, Protocol.pack\_code\_ans(codeAnswer))  
 *# check if code is valid or not and send connection data of users who connected to session back to each other  
 # to start session* if codeAnswer:  
 otherIP = codes.ip\_from\_code(userCode)  
 server.send(clientIP, Protocol.pack\_con\_data(otherIP, "A"))  
 server.send(otherIP, Protocol.pack\_con\_data(clientIP, "H"))  
  
  
def disconnectClient(clientIP, server):  
 *"""  
 Function removes client from users dic and closes the users socket from server  
 :param clientIP: ip of user to disconnect from server  
 :param server: server object to communicate with server  
 """* if clientIP in users.keys():  
 del users[clientIP]  
 server.closeClient(clientIP)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 expired\_q = queue.Queue()  
 codes = Session\_codes(expired\_q)  
 recv\_q = queue.Queue()  
 port = 2000  
 server = Server\_comm(recv\_q, port)  
  
 *# dictionary of opcode and function name called for each opcode* commands = {"00": handle\_login, "01": handle\_signup, "02": handle\_typeUser, "03": handle\_getCode,  
 "04": handle\_codeCheck, "06": handle\_mac\_addr}  
 *# dictionary of users in system, Exp: users{ip : (mac\_addr, userType, username)}* users = {}  
  
 *# start Threads for msgs from server queue and msgs from expired codes queue* threading.Thread(target=handleMsgs, args=(server, recv\_q,)).start()  
 threading.Thread(target=handleExpired, args=(server, expired\_q,)).start()

Client code:

def unpackData(data: str) -> str and list:  
 *"""  
 function gets data and unpacks it based on known opcodes and params protocols  
 :param data: data to unpack  
 :return: the opcode of the data and a list of all it's params  
 """* opcode = data[:2]  
 params = []  
  
 if opcode == "01" or opcode == "02" or opcode == "04":  
 *# 0 if authorized 1 if not* params.append(True) if data[2:] == "0" else params.append(False)  
 elif opcode == "03" or opcode == "00":  
 params.append(data[2:])  
 elif opcode == "05":  
 ip\_len = data[2:4]  
 params.append(data[4:4 + int(ip\_len)])  
 params.append(data[4 + int(ip\_len):])  
  
 return opcode, params  
  
  
def pack\_mac\_addr(macAddr: str) -> str:  
 *"""  
 function packs mac address based on the protocol  
 :param macAddr: computer mac address to send  
 :return: packed data based on protocol  
 """* return "06" + macAddr  
  
  
def pack\_login\_info(username: str, password: str) -> str:  
 *"""  
 function packs login info  
 :param username: username of user to send  
 :param password: password of user to send  
 :return: packed data based on protocol  
 """* return "00" + str(len(username)) + username + password  
  
  
def pack\_signup\_info(username: str, password: str) -> str:  
 *"""  
 function packs signup info  
 :param username: username of user to send  
 :param password: password of user to send  
 :return: packed data based on protocol  
 """* return "01" + str(len(username)) + username + password  
  
  
def pack\_type\_user(typeUser: str) -> str:  
 *"""  
 function packs user type decided based on the protocol  
 :param typeUser: user type wanted, H-Helper A-AssistanceSeeker  
 :return: packed data based on protocol  
 """* msg = "02"  
 if typeUser == "H":  
 msg += "0"  
 elif typeUser == "A":  
 msg += "1"  
 return msg  
  
  
def pack\_code(code: str) -> str:  
 *"""  
 function packs code entered based on the protocol  
 :param code: code entered by user  
 :return: packed data based on protocol  
 """* return "04" + code

import base64  
import secrets  
import socket  
import sys  
import threading  
  
from AsymmetricEncryption import RSA\_cipher  
from SymmetricEncryption import AES\_hash\_cipher  
  
  
*# class to communicate as client with server*class Client\_comm:  
 def \_\_init\_\_(self, serverIP: str, port: int, recv\_q):  
 *"""  
 builder function creates new "Client\_comm" object and runs main loop of client  
 :param serverIP: ip of server to connect to  
 :param port: port in which to connect to server through  
 :param recv\_q: queue of data gotten to server to put  
 """* self.serverIP = serverIP  
 self.port = port  
 self.recv\_q = recv\_q  
 self.socket = socket.socket()  
 self.is\_running = True  
 self.sharedKey = None  
 threading.Thread(target=self.\_main\_loop).start()  
  
 def \_main\_loop(self):  
 *"""  
 main client loop  
 """  
 # connect to server* try:  
 self.socket.connect((self.serverIP, self.port))  
 except Exception as e:  
 sys.exit("server is down, try again later")  
  
 *# create new shared key with server to encrypt and decrypt data* self.\_get\_shared\_key()  
  
 while self.is\_running:  
 *# get datalen from server* try:  
 datalen = self.socket.recv(3).decode()  
 except Exception as e:  
 sys.exit("server is down, try again later")  
  
 if not datalen.isnumeric():  
 sys.exit("server is down, try again later")  
  
 *# get data by data len and decrypt data* try:  
 data = self.socket.recv(int(datalen)).decode()  
 except Exception as e:  
 sys.exit("server is down, try again later")  
  
 data = self.sharedKey.decrypt(data)  
  
 *# put data in recv q for main client* self.recv\_q.put(data)  
  
 def \_get\_shared\_key(self):  
 *"""  
 function creates new shared key and sends it by servers public key to server  
 """  
  
 # get servers string public key for RSA* try:  
 pubKey = self.socket.recv(451).decode()  
 except Exception as e:  
 sys.exit("server is down, try again later")  
 *# check string public key length* if len(pubKey) != 451:  
 sys.exit("server error in key gotten")  
  
 *# create new 64 character random shared key to send to server and for AES encryption* shared\_key\_string = base64.b64encode(secrets.token\_bytes(48)).decode()  
 *# send RSA encrypted shared key to server* try:  
 self.socket.send(RSA\_cipher.encrypt(shared\_key\_string, pubKey))  
 except Exception as e:  
 sys.exit("server is down, try again later")  
  
 *# save AES encryption object of shared key* self.sharedKey = AES\_hash\_cipher(shared\_key\_string)  
  
 def send(self, msg: str):  
 *"""  
 function sends encrypted msg to server  
 :param msg: msg to send to server  
 """  
 # send data only if there is a shared key* while self.sharedKey is None:  
 continue  
  
 *# encrypt msg and send it with msg length* msg = self.sharedKey.encrypt(msg.encode())  
  
 try:  
 self.socket.send((str(len(msg)).zfill(3)).encode() + msg)  
 except Exception as e:  
 print('client comm - send', str(e))  
 sys.exit("server is down, try again later")  
  
 def sendImage(self, data: str, imageData: bytes):  
 *"""  
 Function gets Image data and data header and sends to server  
 :param data: data header to send to server with data about Image  
 :param imageData: data of image  
 """  
 # send data only if there is a shared key* if self.sharedKey is not None:  
 *# encrypt msg and send it with msg length* data = self.sharedKey.encrypt(data.encode())  
 *# send length of data, encrypted data and Image data* try:  
 self.socket.send((str(len(data)).zfill(3)).encode())  
 self.socket.send(data)  
 self.socket.send(imageData)  
 except Exception as e:  
 print('client comm - sendImage', str(e))  
 sys.exit("server is down, try again later")  
  
 def exchangeStatus(self):  
 *"""  
 Function returns True if shared key has already been created and key exchange has completed and False otherwise  
 """* return self.sharedKey is not None  
  
 def close(self):  
 *"""  
 end main loop and close socket  
 """* self.is\_running = False  
 self.socket.close()  
 self.recv\_q.put("close")

import multiprocessing  
import queue  
import threading  
import time  
from uuid import getnode  
  
import wx  
from pubsub import pub  
  
import AssistanceSeeker\_keyboard\_logic  
import AssistanceSeeker\_mouse\_logic  
import AssistanceSeeker\_screen\_logic  
import Client\_protocol as Protocol  
import Helper\_logic  
import Helper\_screen\_logic  
import graphics  
from clientComm import Client\_comm  
  
  
def handleMsgs(client, recv\_q):  
 *"""  
 Function recieves new msgs from queue from server, unpacks the msg based on protocol  
 and calls function by opcode  
 :param client: client object for comm with server  
 :param recv\_q: queue of msgs gotten from server  
 """* while True:  
 data = recv\_q.get()  
 *# if data gotten is close stop getting new msgs* if data == "close":  
 break  
 else:  
 *# unpack data by protocol and call function by opcode from data* opcode, params = Protocol.unpackData(data)  
 print(params)  
 commands[opcode](params)  
  
  
def send\_mac(client):  
 *"""  
 Function gets the computers mac and sends it to server by protocol  
 :param client: client object for comm with server  
 """* mac\_address = ':'.join(['{:02x}'.format((getnode() >> i) & 0xff) for i in range(0, 8 \* 6, 8)][::-1])  
 client.send(Protocol.pack\_mac\_addr(mac\_address))  
  
  
def handle\_login\_ans(params):  
 *"""  
 Function gets the answer of login msg from server and sends the login ans to graphics  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* login\_ans = params[0]  
 wx.CallAfter(pub.sendMessage, "login\_ans", ans=login\_ans)  
  
  
def handle\_signup\_ans(params):  
 *"""  
 Function gets the answer of signup msg from server and sends the signup ans to graphics  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* signup\_ans = params[0]  
 wx.CallAfter(pub.sendMessage, "signup\_ans", ans=signup\_ans)  
  
  
def handle\_typeUser\_ans(params):  
 *"""  
 Function gets the answer of typeUser msg from server and sends the typeUser ans to graphics  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* typeUser\_ans = params[0]  
 wx.CallAfter(pub.sendMessage, "typeUser\_ans", ans=typeUser\_ans)  
  
  
def handle\_getCode\_ans(params):  
 *"""  
 Function gets the answer of getCode msg from server and sends the getCode ans to graphics  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* getCode\_ans = params[0]  
 wx.CallAfter(pub.sendMessage, "gotten\_code", ans=getCode\_ans)  
  
  
def handle\_code\_ans(params):  
 *"""  
 Function gets the answer of code msg from server and sends the code ans to graphics  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* code\_ans = params[0]  
 wx.CallAfter(pub.sendMessage, "code\_ans", ans=code\_ans)  
  
  
def handle\_conData(params):  
 *"""  
 Function gets connection data of other user to connect too and starts process for mouse keyboard and screen  
 accordingly to user type  
 :param params: list of all parameters unpacked and gotten from data from server  
 """* otherIP = params[0]  
 my\_user\_Type = params[1]  
  
 if my\_user\_Type:  
 *# queue for comm between keyboard and client to send msgs when keyboard detects to disconnect from session* close\_queue = multiprocessing.Queue()  
 *# create proccesss for keyboard screen and mouse as both a Helper and Assistance Seeker accordingly  
 # if a user is a Helper send to graphic connecting\_session to switch to connecting session screen* if my\_user\_Type == "H":  
 mouse = multiprocessing.Process(target=Helper\_logic.main\_Helper, args=(otherIP, 2001, None,))  
 keyboard = multiprocessing.Process(target=Helper\_logic.main\_Helper, args=(otherIP, 2002, close\_queue,))  
 screen = multiprocessing.Process(target=Helper\_screen\_logic.main\_Helper\_screen, args=(otherIP,))  
 wx.CallAfter(pub.sendMessage, "connecting\_session")  
 elif my\_user\_Type == "A":  
 mouse = multiprocessing.Process(target=AssistanceSeeker\_mouse\_logic.main\_AS\_mouse, args=(otherIP,))  
 keyboard = multiprocessing.Process(target=AssistanceSeeker\_keyboard\_logic.main\_AS\_keyboard,  
 args=(otherIP, close\_queue,))  
 screen = multiprocessing.Process(target=AssistanceSeeker\_screen\_logic.main\_AS\_screen, args=(otherIP,))  
  
 *# close client to main server (not needed anymore as connection switches to P2P)* client.close()  
  
 *# wait five seconds before starting mouse screen and keyboard process for connecting to session screen  
 # to finish in graphics* time.sleep(5)  
 mouse.start()  
 keyboard.start()  
 screen.start()  
  
 *# call function for when user asks session to close* check\_closed(close\_queue, mouse, screen, keyboard)  
  
  
def check\_closed(close\_queue, mouse, screen, keyboard):  
 *"""  
 Function gets data from close\_queue when data gotten is close  
 (keyboard process tells the client to terminate all processes to end session)  
 terminate all running processes and leave loop  
 :param close\_queue: multiprocessing queue for comm between client and keyboard processes  
 :param mouse: running process of mouse control  
 :param screen: running process of screen control  
 :param keyboard: running process of keyboard control  
 """* while True:  
 data = close\_queue.get()  
 if data == "close":  
 mouse.terminate()  
 screen.terminate()  
 keyboard.terminate()  
 break  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 ip = "10.0.0.38"  
 port = 2000  
 recv\_q = queue.Queue()  
 client = Client\_comm(ip, port, recv\_q)  
  
 *# dictionary of opcode and function name called for each opcode* commands = {"00": handle\_login\_ans, "01": handle\_signup\_ans, "02": handle\_typeUser\_ans,  
 "03": handle\_getCode\_ans, "04": handle\_code\_ans, "05": handle\_conData}  
  
 *# create graphic object* app = wx.App()  
 allGraphics = graphics.MyFrame(client)  
  
 *# start Threads for msgs from server queue and sending computer mac to server* threading.Thread(target=handleMsgs, args=(client, recv\_q,)).start()  
 threading.Thread(target=send\_mac, args=(client,)).start()  
  
 *# run main graphic* app.MainLoop()

import re  
  
import pyperclip  
import wx  
import wx.adv  
from pubsub import pub  
  
import Client\_protocol  
  
  
class MyFrame(wx.Frame):  
 def \_\_init\_\_(self, comm, parent=None):  
 *"""  
 Initialize frame  
 :param comm: client object to comm with server  
 :param parent: parent panel  
 """* super(MyFrame, self).\_\_init\_\_(parent, title="UELP")  
 self.Maximize()  
 self.client = comm  
  
 *# set icon* self.SetIcon(wx.Icon("logo.png", wx.BITMAP\_TYPE\_PNG))  
  
 *# create main panel - to put on the others panels* self.main\_panel = MainPanel(self)  
 box = wx.BoxSizer(wx.VERTICAL)  
 box.Add(self.main\_panel, 1, wx.EXPAND)  
 *# arrange the frame* self.SetSizer(box)  
 self.Layout()  
 self.Show()  
  
  
class MainPanel(wx.Panel):  
 def \_\_init\_\_(self, parent):  
 *"""  
 Initialize Main panel all other panels are put on  
 :param parent: Parent panel  
 """* wx.Panel.\_\_init\_\_(self, parent)  
 self.frame = parent  
 self.SetBackgroundColour("#fcf6f5")  
 v\_box = wx.BoxSizer(wx.VERTICAL)  
 *# create object for each panel to switch between* self.login = LoginPanel(self, self.frame)  
 v\_box.Add(self.login)  
 self.signup = SignUpPanel(self, self.frame)  
 v\_box.Add(self.signup)  
 self.select = SelectUserPanel(self, self.frame)  
 v\_box.Add(self.select)  
 self.helper = HelperPanel(self, self.frame)  
 v\_box.Add(self.helper)  
 self.Assistance\_seeker = ASPanel(self, self.frame)  
 v\_box.Add(self.Assistance\_seeker)  
 self.connecting = ConnectingPanel(self, self.frame)  
 v\_box.Add(self.connecting)  
  
 *# show first panel - login* self.login.Show()  
  
 self.SetSizer(v\_box)  
 self.Layout()  
  
 def change\_screen(self, cur\_screen, screen):  
 *"""  
 Function changes panel currently viewed  
 :param cur\_screen: current panel viewed  
 :param screen: new Panel to view  
 """* cur\_screen.Hide()  
 screen.Show()  
 self.Layout()  
  
  
class LoginPanel(wx.Panel):  
 def \_\_init\_\_(self, parent, frame):  
 *"""  
 Login screen panel  
 :param parent: Parent panel  
 :param frame: Frame parent  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, style=wx.SIMPLE\_BORDER, size=(1920, 1080))  
 self.frame = frame  
 self.parent = parent  
 self.SetBackgroundColour("#fdf0d0")  
  
 sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# initialize title on panel* title = wx.StaticText(self, -1, label="UELP")  
 titlefont = wx.Font(68, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Algerian")  
 title.SetForegroundColour("#aa7c57")  
 title.SetFont(titlefont)  
  
 *# initialize about button on panel* self.about = wx.StaticBitmap(self, wx.ID\_ANY, wx.Bitmap("info.png"), pos=(1850, 5))  
 self.about.Bind(wx.EVT\_LEFT\_DOWN, self.show\_info\_dialog)  
  
 *# initialize signin title on panel* signin = wx.StaticText(self, -1, label="Sign In", pos=(720, 100))  
 signinFont = wx.Font(125, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 signin.SetFont(signinFont)  
 signin.SetForegroundColour("#3f4043")  
  
 *# initialize username textctrl on panel* self.nameField = wx.TextCtrl(self, -1, style=wx.TE\_PROCESS\_ENTER, size=(600, 100))  
 self.nameField.SetFont(wx.Font(65, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
 self.set\_placeholder\_text(self.nameField, "Username")  
  
 *# Add username image to the left of the nameField* wx.StaticBitmap(self, -1, wx.Bitmap("user.png"), pos=(550, 365))  
  
 *# bind text change event to function for username textctrl* self.nameField.Bind(wx.EVT\_TEXT, self.on\_text\_change)  
  
 *# initialize password textctrl on panel* self.passField = wx.TextCtrl(self, -1, style=wx.TE\_PASSWORD, size=(600, 100))  
 self.passField.SetFont(wx.Font(50, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
 self.set\_placeholder\_text(self.passField, "a12fgtegp")  
  
 *# Add password image to the left of the nameField* wx.StaticBitmap(self, -1, wx.Bitmap("pass.png"), pos=(550, 502))  
  
 *# bind text change event to function for password textctrl* self.passField.Bind(wx.EVT\_TEXT, self.on\_text\_change)  
  
 *# initialize login button bitmaps and the static bitmap acting as the button* self.login\_bitmap = wx.Bitmap("login.png")  
 self.login\_off\_bitmap = wx.Bitmap("login\_off.png")  
 self.next = wx.StaticBitmap(self, wx.ID\_ANY, self.login\_off\_bitmap)  
 self.next.Bind(wx.EVT\_LEFT\_DOWN, self.on\_next\_click)  
 self.can\_press = False  
  
 *# set sizer for switching to signup panel elements* signup\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 *# initialize no account text to panel* no\_account = wx.StaticText(self, -1, label="Dont have an account?")  
 no\_account\_font = wx.Font(18, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT")  
 no\_account.SetFont(no\_account\_font)  
  
 *# initialize signup text so can be pressed to switch to signup panel* self.signup = wx.StaticText(self, -1, label="Sign Up")  
 signup\_font = wx.Font(18, wx.DEFAULT, wx.NORMAL, wx.NORMAL, True, "Calisto MT")  
 self.signup.Bind(wx.EVT\_LEFT\_DOWN, self.handle\_signup\_screen)  
 self.signup.SetFont(signup\_font)  
 self.signup.SetForegroundColour("#0000FF")  
  
 *# add all text to signup sizer* signup\_sizer.Add(no\_account, 0, wx.ALL, 5)  
 signup\_sizer.AddSpacer(8)  
 signup\_sizer.Add(self.signup, 0, wx.ALL, 5)  
  
 *# create list of all interactive elemnts to disable/enable when needed* self.interactive\_elements = [self.nameField, self.passField, self.next]  
  
 *# Add all elements to sizer* sizer.Add(title, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(260)  
 sizer.Add(self.nameField, 0, wx.CENTER, 5)  
 sizer.AddSpacer(35)  
 sizer.Add(self.passField, 0, wx.CENTER, 5)  
 sizer.AddSpacer(110)  
 sizer.Add(self.next, 0, wx.CENTER, 5)  
 sizer.Add(signup\_sizer, 0, wx.CENTER, 5)  
  
 *# Add logo at the bottom right of screen* wx.StaticBitmap(self, -1, wx.Bitmap("logo.png"), pos=(1680, 920))  
  
 *# Arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
  
 *# invalidText shown when gotten negetive answer and need to show on screen* self.invalidText = None  
  
 pub.subscribe(self.handle\_login\_ans, "login\_ans")  
  
 def handle\_login\_ans(self, ans):  
 *"""  
 Function called when gotten login request ans from server, act accordingly  
 (change screen or let user know login incorrect)  
 :param ans: answer to login request user sent  
 """* if ans == "1":  
 self.show\_invalid\_message("Username or password incorrect", 2000)  
 elif ans == "2":  
 self.show\_invalid\_message("User is logged in", 2000)  
 elif ans == "0":  
 self.parent.change\_screen(self, self.parent.select)  
  
 def show\_invalid\_message(self, msg, time):  
 *"""  
 Function shows invalid message on screen for certain time  
 :param msg: msg to show on screen  
 :param time: amount of time to show message on screen  
 """  
 # Set invalid text as msg gotten* self.invalidText = wx.StaticText(self, label=msg, style=wx.ALIGN\_CENTER)  
 self.invalidText.SetForegroundColour(wx.RED)  
 self.invalidText.SetFont(wx.Font(18, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
  
 *# place invalid text in right position in sizer* sizer = self.GetSizer()  
 sizer.Insert(sizer.GetItemCount(), self.invalidText, 0, wx.ALIGN\_TOP | wx.CENTER, 10)  
  
 *# disable interactive elements so user has to view invalid text for certain time* self.disable\_interactive\_elements()  
 self.Layout()  
  
 timer = wx.Timer(self)  
 self.Bind(wx.EVT\_TIMER, self.on\_timer, timer)  
 *# start timer for certain time gotten* timer.Start(time, oneShot=True)  
  
 def on\_timer(self, event):  
 *"""  
 Function called when timer ends and erases invalid text currently seen  
 :param event: event for time  
 """  
 # Remove the invalid message if still exists* if self.invalidText is not None:  
 self.invalidText.Destroy()  
 self.Layout()  
 self.enable\_interactive\_elements()  
  
 def set\_placeholder\_text(self, text\_ctrl, placeholder):  
 *"""  
 Function called to set certain placeholder text in textctrl and bind focus and unfocus events for textctrl  
 :param text\_ctrl: text\_ctrl to place text in  
 :param placeholder: text to place in text ctrl  
 """* text\_ctrl.SetValue(placeholder)  
 *# Set text color to grey* text\_ctrl.SetForegroundColour('#808080')  
 *# Reset text style* text\_ctrl.SetDefaultStyle(wx.TextAttr(wx.NullColour, wx.NullColour))  
  
 *# bind focus events to textctrl* text\_ctrl.Bind(wx.EVT\_SET\_FOCUS, self.on\_set\_focus)  
 text\_ctrl.Bind(wx.EVT\_KILL\_FOCUS, self.on\_kill\_focus)  
  
 def on\_set\_focus(self, event):  
 *"""  
 Function called when text ctrl is focused on, if placeholder texts for textctrls are there then  
 empty text ctrl and change text color  
 :param event: event of focus  
 """* ctrl = event.GetEventObject()  
  
 if ctrl.GetValue() == "Username" or ctrl.GetValue() == "a12fgtegp":  
 ctrl.SetValue("")  
 ctrl.SetForegroundColour(wx.BLACK)  
 self.Layout()  
 event.Skip()  
  
 def on\_kill\_focus(self, event):  
 *"""  
 Function called when text ctrl is not focused on, if text ctrl is empty set text in textctrl to placeholder  
 text and change text color  
 :param event: event of kill focus  
 """* ctrl = event.GetEventObject()  
  
 if ctrl.IsEmpty():  
 if ctrl == self.nameField:  
 ctrl.SetValue("Username")  
 elif ctrl == self.passField:  
 ctrl.SetValue("a12fgtegp")  
 ctrl.SetForegroundColour('#808080') *# Set text color to grey* event.Skip()  
  
 def on\_text\_change(self, event):  
 *"""  
 Function called when text in text ctrl is changed and checks if text fit criteria for pressing/not pressing  
 login button and changes boolean value accordingly  
 :param event: event of text change  
 """* username = self.nameField.GetValue()  
 password = self.passField.GetValue()  
  
 *# if text in username and password textctrl isn't placeholder text and both are not empty can press button* if not username or not password or username == "Username" or password == "a12fgtegp":  
 *# change bitmap of button and boolean value if can press button* self.next.SetBitmap(self.login\_off\_bitmap)  
 self.can\_press = False  
 elif not self.can\_press:  
 self.next.SetBitmap(self.login\_bitmap)  
 self.can\_press = True  
  
 def on\_next\_click(self, event):  
 *"""  
 Function called when login button pressed, function checks if user can press button  
 and packs username and password gotten from user by protocol and sends to server  
 :param event: event of button click  
 """* username = self.nameField.GetValue()  
 password = self.passField.GetValue()  
  
 if self.can\_press:  
 msg2send = Client\_protocol.pack\_login\_info(username, password)  
 self.frame.client.send(msg2send)  
  
 def show\_info\_dialog(self, event):  
 *"""  
 Function called when info button pressed, show info on system  
 :param event: event of info click  
 """* info = wx.adv.AboutDialogInfo()  
 info.SetName("Itamar system")  
 info.SetDescription("This system allow you to Control and help friends computer from your own")  
 info.SetCopyright("(C) 2024-2030")  
 info.AddDeveloper("Itamar Nahum")  
 info.AddArtist("Itamar Nahum")  
 info.AddDocWriter("Itamar Nahum")  
 info.SetVersion("1.0")  
 info.SetName("UELP")  
 info.SetLicence("Atid \n Cramim \n Israel")  
 wx.adv.AboutBox(info)  
  
 def handle\_signup\_screen(self, event):  
 *"""  
 Function called when signup text pressed, change screen to signup panel  
 :param event: event of signup screen click  
 """* self.parent.change\_screen(self, self.parent.signup)  
  
 def \_check\_password(self, password):  
 *"""  
 Function checks password user entered  
 :param password: password gotten from user  
 """* pattern = r'[^a-zA-Z0-9\s]'  
 is\_ok = True  
 *# password must have at least 8 characters one uppercase letter and one special character,  
 # show invalid message accordingly* if len(password) < 8:  
 self.show\_invalid\_message("Password should be 8 characters and up", 2000)  
 is\_ok = False  
 elif not re.search(pattern, password):  
 self.show\_invalid\_message("Password must include one special character", 2000)  
 is\_ok = False  
 elif password == password.lower():  
 self.show\_invalid\_message("Password must include one uppercase letter", 2000)  
 is\_ok = False  
  
 return is\_ok  
  
 def disable\_interactive\_elements(self):  
 *"""  
 Function disables all interactive elements in interactive elements list  
 """* for element in self.interactive\_elements:  
 element.Disable()  
  
 def enable\_interactive\_elements(self):  
 *"""  
 Function enables all interactive elements in interactive elements list  
 """* for element in self.interactive\_elements:  
 element.Enable()  
  
  
class SignUpPanel(wx.Panel):  
 def \_\_init\_\_(self, parent, frame):  
 *"""  
 Signup screen panel  
 :param parent: parent panel  
 :param frame: frame parent  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, style=wx.SIMPLE\_BORDER, size=(1920, 1080))  
 self.frame = frame  
 self.parent = parent  
 self.SetBackgroundColour("#fdf0d0")  
 sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# initialize title on panel* title = wx.StaticText(self, -1, label="UELP")  
 titlefont = wx.Font(68, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Algerian")  
 title.SetForegroundColour("#aa7c57")  
 title.SetFont(titlefont)  
  
 *# initialize signup title on panel* signup = wx.StaticText(self, -1, label="Sign Up", pos=(705, 100))  
 signupFont = wx.Font(125, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 signup.SetFont(signupFont)  
 signup.SetForegroundColour("#3f4043")  
  
 *# initialize username textctrl on panel* self.nameField = wx.TextCtrl(self, -1, style=wx.TE\_PROCESS\_ENTER, size=(600, 100))  
 self.nameField.SetFont(wx.Font(65, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
 self.set\_placeholder\_text(self.nameField, "Username")  
  
 *# Add username image to the left of the nameField* wx.StaticBitmap(self, -1, wx.Bitmap("user.png"), pos=(550, 365))  
  
 *# bind text change event to function for username textctrl* self.nameField.Bind(wx.EVT\_TEXT, self.on\_text\_change)  
  
 *# initialize password textctrl on panel* self.passField = wx.TextCtrl(self, -1, style=wx.TE\_PASSWORD, size=(600, 100))  
 self.passField.SetFont(wx.Font(50, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
 self.set\_placeholder\_text(self.passField, "a12fgtegp")  
  
 *# Add password image to the left of the nameField* wx.StaticBitmap(self, -1, wx.Bitmap("pass.png"), pos=(550, 502))  
  
 *# bind text change event to function for password textctrl* self.passField.Bind(wx.EVT\_TEXT, self.on\_text\_change) *# Bind text change event  
  
 # initialize signup button bitmaps and the static bitmap acting as the button* self.signup\_bitmap = wx.Bitmap("signup.png")  
 self.signup\_off\_bitmap = wx.Bitmap("signup\_off.png")  
 self.next = wx.StaticBitmap(self, wx.ID\_ANY, wx.Bitmap("signup\_off.png"))  
 self.next.Bind(wx.EVT\_LEFT\_DOWN, self.on\_next\_click)  
 self.can\_press = False  
  
 *# set sizer for switching to signin panel elements* signin\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 *# initialize have account text to panel* have\_account = wx.StaticText(self, -1, label="Already have an account?")  
 have\_account\_font = wx.Font(18, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT")  
 have\_account.SetFont(have\_account\_font)  
  
 *# initialize text acting as button to switch to login panel* self.login = wx.StaticText(self, -1, label="Login")  
 signin\_font = wx.Font(18, wx.DEFAULT, wx.NORMAL, wx.NORMAL, True, "Calisto MT")  
 self.login.Bind(wx.EVT\_LEFT\_DOWN, self.handle\_signin\_screen)  
 self.login.SetFont(signin\_font)  
 self.login.SetForegroundColour("#0000FF")  
  
 *# add elements to sizer* signin\_sizer.Add(have\_account, 0, wx.ALL, 5)  
 signin\_sizer.AddSpacer(8)  
 signin\_sizer.Add(self.login, 0, wx.ALL, 5)  
  
 *# create list of all interactive elemnts to disable/enable when needed* self.interactive\_elements = [self.nameField, self.passField, self.next]  
  
 *# Add all elements to sizer* sizer.Add(title, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(260)  
 sizer.Add(self.nameField, 0, wx.CENTER, 5)  
 sizer.AddSpacer(35)  
 sizer.Add(self.passField, 0, wx.CENTER, 5)  
 sizer.AddSpacer(110)  
 sizer.Add(self.next, 0, wx.CENTER, 5)  
 sizer.Add(signin\_sizer, 0, wx.CENTER, 5)  
  
 *# Add logo at the bottom right of panel* wx.StaticBitmap(self, -1, wx.Bitmap("logo.png"), pos=(1680, 920))  
  
 *# Arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
  
 *# invalidText shown when gotten negetive answer and need to show on screen* self.invalidText = None  
  
 pub.subscribe(self.handle\_signup\_ans, "signup\_ans")  
  
 def handle\_signup\_ans(self, ans):  
 *"""  
 Function called when gotten signup request ans from server, act accordingly  
 (change screen or let user know signup incorrect)  
 :param ans: answer to signup request user sent  
 """* if not ans:  
 self.show\_invalid\_message("Username already taken", 2000)  
 else:  
 self.parent.change\_screen(self, self.parent.select)  
  
 def show\_invalid\_message(self, msg, time):  
 *"""  
 Function shows invalid message on screen for certain time  
 :param msg: msg to show on screen  
 :param time: amount of time to show message on screen  
 """  
 # Set invalid text as msg gotten* self.invalidText = wx.StaticText(self, label=msg, style=wx.ALIGN\_CENTER)  
 self.invalidText.SetForegroundColour(wx.RED)  
 self.invalidText.SetFont(wx.Font(18, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
  
 *# place invalid text in right position in sizer* sizer = self.GetSizer()  
 sizer.Insert(sizer.GetItemCount(), self.invalidText, 0, wx.ALIGN\_TOP | wx.CENTER, 10)  
  
 *# disable interactive elements so user has to view invalid text for certain time* self.disable\_interactive\_elements()  
 self.Layout()  
  
 timer = wx.Timer(self)  
 self.Bind(wx.EVT\_TIMER, self.on\_timer, timer)  
 *# start timer for certain time gotten* timer.Start(time, oneShot=True)  
  
 def on\_timer(self, event):  
 *"""  
 Function called when timer ends and erases invalid text currently seen  
 :param event: event for time  
 """  
 # Remove the invalid message if still exists* if self.invalidText is not None:  
 self.invalidText.Destroy()  
 self.Layout()  
 self.enable\_interactive\_elements()  
  
 def set\_placeholder\_text(self, text\_ctrl, placeholder):  
 *"""  
 Function called to set certain placeholder text in textctrl and bind focus and unfocus events for textctrl  
 :param text\_ctrl: text\_ctrl to place text in  
 :param placeholder: text to place in text ctrl  
 """* text\_ctrl.SetValue(placeholder)  
  
 *# Set text color to grey* text\_ctrl.SetForegroundColour('#808080')  
  
 *# Reset text style* text\_ctrl.SetDefaultStyle(wx.TextAttr(wx.NullColour, wx.NullColour))  
  
 *# bind focus events to textctrl* text\_ctrl.Bind(wx.EVT\_SET\_FOCUS, self.on\_set\_focus)  
 text\_ctrl.Bind(wx.EVT\_KILL\_FOCUS, self.on\_kill\_focus)  
  
 def on\_set\_focus(self, event):  
 *"""  
 Function called when text ctrl is focused on, if placeholder texts for textctrls are there then  
 empty text ctrl and change text color  
 :param event: event of focus  
 """* ctrl = event.GetEventObject()  
  
 if ctrl.GetValue() == "Username" or ctrl.GetValue() == "a12fgtegp":  
 ctrl.SetValue("")  
 ctrl.SetForegroundColour(wx.BLACK)  
 self.Layout()  
 event.Skip()  
  
 def on\_kill\_focus(self, event):  
 *"""  
 Function called when text ctrl is not focused on, if text ctrl is empty set text in textctrl to placeholder  
 text and change text color  
 :param event: event of kill focus  
 """* ctrl = event.GetEventObject()  
  
 if ctrl.IsEmpty():  
 if ctrl == self.nameField:  
 ctrl.SetValue("Username")  
 elif ctrl == self.passField:  
 ctrl.SetValue("a12fgtegp")  
 *# Set text color to grey* ctrl.SetForegroundColour('#808080')  
 event.Skip()  
  
 def on\_text\_change(self, event):  
 *"""  
 Function called when text in text ctrl is changed and checks if text fit criteria for pressing/not pressing  
 signup button and changes boolean value accordingly  
 :param event: event of text change  
 """* username = self.nameField.GetValue()  
 password = self.passField.GetValue()  
  
 *# if text in username and password textctrl isn't placeholder text and both are not empty can press button* if not username or not password or username == "Username" or password == "a12fgtegp":  
 *# change bitmap of button and boolean value if can press button* self.next.SetBitmap(self.signup\_off\_bitmap)  
 self.can\_press = False  
 elif not self.can\_press:  
 self.next.SetBitmap(self.signup\_bitmap)  
 self.can\_press = True  
  
 def on\_next\_click(self, event):  
 *"""  
 Function called when signup button pressed, function checks if user can press button  
 and packs username and password gotten from user by protocol and sends to server  
 :param event: event of button click  
 """* username = self.nameField.GetValue()  
 password = self.passField.GetValue()  
  
 if self.can\_press and self.\_check\_password(password):  
 msg2send = Client\_protocol.pack\_signup\_info(username, password)  
 self.frame.client.send(msg2send)  
  
 def handle\_signin\_screen(self, event):  
 *"""  
 Function called when signin text pressed, change screen to signup panel  
 :param event: event of signup screen click  
 """* self.parent.change\_screen(self, self.parent.login)  
  
 def \_check\_password(self, password):  
 *"""  
 Function checks password user entered  
 :param password: password gotten from user  
 """* pattern = r'[^a-zA-Z0-9\s]'  
 is\_ok = True  
  
 *# password must have at least 8 characters one uppercase letter and one special character,  
 # show invalid message accordingly* if len(password) < 8:  
 self.show\_invalid\_message("Password should be 8 characters and up", 2000)  
 is\_ok = False  
 elif not re.search(pattern, password):  
 self.show\_invalid\_message("Password must include one special character", 2000)  
 is\_ok = False  
 elif password == password.lower():  
 self.show\_invalid\_message("Password must include one uppercase letter", 2000)  
 is\_ok = False  
  
 return is\_ok  
  
 def disable\_interactive\_elements(self):  
 *"""  
 Function disables all interactive elements in interactive elements list  
 """* for element in self.interactive\_elements:  
 element.Disable()  
  
 def enable\_interactive\_elements(self):  
 *"""  
 Function enables all interactive elements in interactive elements list  
 """* for element in self.interactive\_elements:  
 element.Enable()  
  
  
class SelectUserPanel(wx.Panel):  
 def \_\_init\_\_(self, parent, frame):  
 *"""  
 Select user panel  
 :param parent: Parent panel  
 :param frame: frame parent  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, style=wx.SIMPLE\_BORDER, size=(1920, 1080))  
 self.frame = frame  
 self.parent = parent  
 self.SetBackgroundColour("#fdf0d0")  
  
 sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# initialize title on panel* title = wx.StaticText(self, -1, label="UELP")  
 titlefont = wx.Font(68, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Algerian")  
 title.SetForegroundColour("#aa7c57")  
 title.SetFont(titlefont)  
  
 *# Initialize select user text on panel* select = wx.StaticText(self, -1, label="Select User Type")  
 selectfont = wx.Font(55, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 select.SetFont(selectfont)  
 select.SetForegroundColour("#3f4043")  
  
 *# Initialize sizer for Helper & Assistance Seeker buttons* btnBox = wx.BoxSizer(wx.HORIZONTAL)  
  
 *# Initialize helper static bitmap button on panel, bind to hover and pressing events* self.helper = wx.StaticBitmap(self, -1, wx.Bitmap("Helper\_choose.png"))  
 self.helper.Bind(wx.EVT\_ENTER\_WINDOW, self.on\_hover)  
 self.helper.Bind(wx.EVT\_LEAVE\_WINDOW, self.on\_leave)  
 self.helper.Bind(wx.EVT\_LEFT\_DOWN, self.handle\_helper)  
  
 *# Mark this bitmap as window variant to differ helper from Assistance Seeker button* self.helper.SetWindowVariant(wx.WINDOW\_VARIANT\_SMALL)  
  
 *# Initialize Assistance Seeker static bitmap button on panel, bind to hover and pressing events* self.As = wx.StaticBitmap(self, -1, wx.Bitmap("assistance\_seeker.png"))  
 self.As.Bind(wx.EVT\_ENTER\_WINDOW, self.on\_hover)  
 self.As.Bind(wx.EVT\_LEAVE\_WINDOW, self.on\_leave)  
 self.As.Bind(wx.EVT\_LEFT\_DOWN, self.handle\_AS) *# Bind left click event  
 # Mark this bitmap as window variant to differ Assistance seeker from Helper button* self.As.SetWindowVariant(wx.WINDOW\_VARIANT\_NORMAL)  
  
 *# Add elements to sizer* btnBox.Add(self.helper, 0, wx.Center, 5)  
 btnBox.AddSpacer(150)  
 btnBox.Add(self.As, 0, wx.Center)  
  
 *# Add all elements to sizer* sizer.Add(title, 0, wx.CENTER | wx.TOP, 5)  
 sizer.Add(select, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(20)  
 sizer.Add(btnBox, wx.CENTER | wx.ALL, 5)  
  
 *# Add logo at the bottom right of panel* wx.StaticBitmap(self, -1, wx.Bitmap("logo.png"), pos=(1680, 920))  
  
 *# Arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
 self.invalidText = None  
 self.userType = None  
  
 pub.subscribe(self.handle\_typeUser\_ans, "typeUser\_ans")  
  
 def on\_hover(self, event):  
 *"""  
 Function called when button hovered on, change bitmap of button accordingly to hover bitmap  
 :param event: event of button of hover  
 """* bitmap = event.GetEventObject()  
 *# check which button gotten, AS or Helper* if bitmap.GetWindowVariant() == wx.WINDOW\_VARIANT\_SMALL:  
 bitmap.SetBitmap(wx.Bitmap("Helper\_choose\_hover.png"))  
 else:  
 bitmap.SetBitmap(wx.Bitmap("assistance\_seeker\_hover.png"))  
 event.Skip()  
  
 def on\_leave(self, event):  
 *"""  
 Function called when button stops hovered on, change bitmap of button accordingly to normal bitmap  
 :param event: event of button of hover  
 """* bitmap = event.GetEventObject()  
 *# check which button gotten, AS or Helper* if bitmap.GetWindowVariant() == wx.WINDOW\_VARIANT\_SMALL:  
 bitmap.SetBitmap(wx.Bitmap("Helper\_choose.png"))  
 else:  
 bitmap.SetBitmap(wx.Bitmap("assistance\_seeker.png"))  
 event.Skip()  
  
 def handle\_typeUser\_ans(self, ans):  
 *"""  
 Function called when gotten type user request ans from server, act accordingly  
 (change screen or let user know user type pressed invalid based on user type pressed)  
 :param ans: answer to type user request user sent  
 """* if self.userType == "H":  
 if not ans:  
 self.show\_invalid\_message("You do not have Helper permissions", 2000)  
 else:  
 *# when changing to helper screen send opcode for getting new code request to server* self.parent.change\_screen(self, self.parent.helper)  
 self.frame.client.send("03")  
  
 elif self.userType == "A":  
 if not ans:  
 self.show\_invalid\_message("User type choice is not valid", 2000)  
 else:  
 self.parent.change\_screen(self, self.parent.Assistance\_seeker)  
  
 def handle\_helper(self, event):  
 *"""  
 Function called when helper button pressed, send user type request based on protocol and set user Type asked for  
 :param event: event of button pressed  
 """* self.userType = "H"  
 msg2send = Client\_protocol.pack\_type\_user(self.userType)  
 self.frame.client.send(msg2send)  
  
 def handle\_AS(self, event):  
 *"""  
 Function called when AS button pressed, send user type request based on protocol and set user Type asked for  
 :param event: event of button pressed  
 """* self.userType = "A"  
 msg2send = Client\_protocol.pack\_type\_user(self.userType)  
 self.frame.client.send(msg2send)  
  
 def show\_invalid\_message(self, msg, time):  
 *"""  
 Function shows invalid message on screen for certain time  
 :param msg: msg to show on screen  
 :param time: amount of time to show message on screen  
 """  
 # Set invalid text as msg gotten* self.invalidText = wx.StaticText(self, label=msg, style=wx.ALIGN\_CENTER)  
 invalidTextfont = wx.Font(48, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, False, "Calisto MT")  
 self.invalidText.SetFont(invalidTextfont)  
 self.invalidText.SetForegroundColour(wx.RED)  
  
 *# set invalid text position and hide buttons to press* self.invalidText.SetPosition((480, 540))  
  
 *# hide both buttons to show invalid message* self.As.Hide()  
 self.helper.Hide()  
 self.Layout()  
  
 *# set user type to be None as user type asked for was not gotten* self.userType = None  
  
 timer = wx.Timer(self)  
 self.Bind(wx.EVT\_TIMER, self.on\_timer, timer)  
 *# start timer for certain time gotten* timer.Start(time, oneShot=True)  
  
 def on\_timer(self, event):  
 *"""  
 Function called when timer ends and erases invalid text currently seen  
 :param event: event for time  
 """  
 # Remove the invalid message if still exists and show buttons again* if self.invalidText is not None:  
 self.invalidText.Destroy()  
 self.As.Show()  
 self.helper.Show()  
 self.Layout()  
  
  
class HelperPanel(wx.Panel):  
 def \_\_init\_\_(self, parent, frame):  
 *"""  
 Helper user panel  
 :param parent: Parent panel  
 :param frame: frame parent  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, style=wx.SIMPLE\_BORDER, size=(1920, 1080))  
 self.frame = frame  
 self.parent = parent  
 self.SetBackgroundColour("#fdf0d0")  
  
 sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# Initialize title on panel* title = wx.StaticText(self, -1, label="UELP")  
 titlefont = wx.Font(68, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Algerian")  
 title.SetForegroundColour("#aa7c57")  
 title.SetFont(titlefont)  
  
 *# Initialize your code text on panel* yourcode = wx.StaticText(self, -1, label="Your Code")  
 yourcodefont = wx.Font(120, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 yourcode.SetFont(yourcodefont)  
 yourcode.SetForegroundColour("#3f4043")  
  
 *# Initialize Code static text on panel (Set as hidden until gotten code from server)* self.code = wx.StaticText(self, -1, label="None")  
 codefont = wx.Font(150, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT")  
 self.code.SetForegroundColour("#000000")  
 self.code.SetFont(codefont)  
 self.code.Hide()  
  
 *# Initialize static Bitmap button of copying code to clipboard, bind to hover and clicking events* self.copy\_code\_bitmap = wx.Bitmap("copy\_code.png")  
 self.copy\_code\_hover\_bitmap = wx.Bitmap("copy\_code\_hover.png")  
 self.copy\_code = wx.StaticBitmap(self, -1, self.copy\_code\_bitmap)  
 self.copy\_code.Bind(wx.EVT\_ENTER\_WINDOW, self.on\_hover)  
 self.copy\_code.Bind(wx.EVT\_LEAVE\_WINDOW, self.on\_leave)  
 self.copy\_code.Bind(wx.EVT\_LEFT\_DOWN, self.handle\_copied)  
  
 *# Add all elements to sizer* sizer.Add(title, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(50)  
 sizer.Add(yourcode, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(100)  
 sizer.Add(self.code, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(100)  
 sizer.Add(self.copy\_code, 0, wx.CENTER, 5)  
  
 *# Add logo at the bottom right of panel* wx.StaticBitmap(self, -1, wx.Bitmap("logo.png"), pos=(1680, 920))  
  
 *# Arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
  
 pub.subscribe(self.handle\_code\_gotten, "gotten\_code")  
 pub.subscribe(self.handle\_connecting\_session, "connecting\_session")  
  
 def handle\_code\_gotten(self, ans):  
 *"""  
 Function called when gotten code request answer from server, act accordingly  
 (ask user if wants new code or show code)  
 :param ans: answer to code request (code or expired code opcode)  
 """  
  
 # answer gotten is either expired code opcode ot just the code gotten* if ans == "2":  
 result = wx.MessageDialog(None, "Your code expired!\nWould you like a new code?", "Code Expiration",  
 wx.YES\_NO | wx.ICON\_QUESTION | wx.CENTRE)  
 *# Check the user's response to expired code message, if wants new code send a new request  
 # if not take user back to select screen* if result == wx.YES:  
 self.frame.client.send("03")  
 else:  
 self.parent.change\_screen(self, self.parent.select)  
 else:  
 *# set session code element as code gotten and show code* self.code.SetLabel(ans)  
 self.code.Show()  
 self.Layout()  
  
 def on\_hover(self, event):  
 *"""  
 Function called when button hovered on, change bitmap of button accordingly to hover bitmap  
 :param event: event of button of hover  
 """* self.copy\_code.SetBitmap(self.copy\_code\_hover\_bitmap)  
  
 def on\_leave(self, event):  
 *"""  
 Function called when button not hovered on, change bitmap of button accordingly to regular bitmap  
 :param event: event of button of hover  
 """* self.copy\_code.SetBitmap(self.copy\_code\_bitmap)  
  
 def handle\_copied(self, event):  
 *"""  
 Function called when copy button is pressed, change bitmap of button,  
 disable button and copy code of session to clipboard  
 :param event: event of button press  
 """* self.copy\_code.SetBitmap(wx.Bitmap("copied.png"))  
 *# copy Code to clipboard* pyperclip.copy(self.code.GetLabel())  
  
 *# unbind all copy code events* self.copy\_code.Unbind(wx.EVT\_ENTER\_WINDOW)  
 self.copy\_code.Unbind(wx.EVT\_LEAVE\_WINDOW)  
 self.copy\_code.Unbind(wx.EVT\_LEFT\_DOWN)  
  
 def handle\_connecting\_session(self):  
 *"""  
 Function is called from client logic when other user connected to session, switch to connecting screen panel  
 and start the animation of dots  
 """* self.parent.change\_screen(self, self.parent.connecting)  
 self.parent.connecting.start\_dots\_animation()  
  
  
class ASPanel(wx.Panel):  
 def \_\_init\_\_(self, parent, frame):  
 *"""  
 Helper user panel  
 :param parent: Parent panel  
 :param frame: frame parent  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, style=wx.SIMPLE\_BORDER, size=(1920, 1080))  
 self.frame = frame  
 self.parent = parent  
 self.SetBackgroundColour("#fdf0d0")  
  
 sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# Initialize title on panel* title = wx.StaticText(self, -1, label="UELP")  
 titlefont = wx.Font(68, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Algerian")  
 title.SetForegroundColour("#aa7c57")  
 title.SetFont(titlefont)  
  
 *# Initialize enter code text on panel* enter = wx.StaticText(self, -1, label="Enter Session Code")  
 enterfont = wx.Font(75, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 enter.SetForegroundColour("#3f4043")  
 enter.SetFont(enterfont)  
  
 *# Initialize textctrl to enter code on panel, bind to text changing event* self.codeField = wx.TextCtrl(self, -1, style=wx.TE\_PROCESS\_ENTER, size=(800, -1))  
 codefont = wx.Font(90, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT")  
 self.codeField.SetFont(codefont)  
 self.codeField.Bind(wx.EVT\_TEXT, self.on\_text\_change)  
  
 *# Initialize static bitmap button to connect to session, bind to click event* self.connect = wx.StaticBitmap(self, wx.ID\_ANY, wx.Bitmap("connect\_off.png"))  
 self.connect.Bind(wx.EVT\_LEFT\_DOWN, self.on\_connect)  
 self.can\_press = False  
  
 *# Add all elements to sizer* sizer.Add(title, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(75)  
 sizer.Add(enter, 0, wx.CENTER | wx.TOP, 5)  
 sizer.AddSpacer(100)  
 sizer.Add(self.codeField, 0, wx.CENTER | wx.ALL, 5)  
 sizer.AddSpacer(180)  
 sizer.Add(self.connect, 0, wx.ALIGN\_CENTER\_HORIZONTAL, 5)  
  
 *# Add logo at the bottom right of panel* wx.StaticBitmap(self, -1, wx.Bitmap("logo.png"), pos=(1680, 920))  
  
 *# arrange the screen* self.SetSizer(sizer)  
 self.invalidText = None  
 self.Layout()  
 self.Hide()  
  
 pub.subscribe(self.handle\_code\_ans, "code\_ans")  
  
 def handle\_code\_ans(self, ans):  
 *"""  
 Function called when gotten code entered answer from server, act accordingly  
 (switch to connecting session panel or show invalid message)  
 :param ans: answer to code entered (code valid or invalid)  
 """* if ans:  
 self.parent.change\_screen(self, self.parent.connecting)  
 self.parent.connecting.start\_dots\_animation()  
 else:  
 self.show\_invalid\_message("Session code incorrect", 2000)  
  
 def on\_text\_change(self, event):  
 *"""  
 Function called when text in textctrl changes, checks if text is empty or isn't and changes bitmap of button and  
 can\_press boolean value accordingly  
 :param event: event of textctrl value change  
 """* text = self.codeField.GetValue()  
  
 if text != "":  
 if not self.can\_press:  
 self.connect.SetBitmap(wx.Bitmap("connect.png"))  
 self.can\_press = True  
 else:  
 self.connect.SetBitmap(wx.Bitmap("connect\_off.png"))  
 self.can\_press = False  
  
 def on\_connect(self, event):  
 *"""  
 Function called when connect button is pressed, check if boolean value of can\_press is True  
 and send code in textctrl field by protocol to server  
 :param event: event of connect button pressed  
 """* code\_in\_text = self.codeField.GetValue()  
  
 if self.can\_press:  
 msg2send = Client\_protocol.pack\_code(code\_in\_text)  
 self.frame.client.send(msg2send)  
  
 def show\_invalid\_message(self, msg, time):  
 *"""  
 Function shows invalid message on screen for certain time  
 :param msg: msg to show on screen  
 :param time: amount of time to show message on screen  
 """  
 # Set invalid text as msg gotten* self.invalidText = wx.StaticText(self, label=msg, style=wx.ALIGN\_CENTER)  
 self.invalidText.SetFont(wx.Font(35, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Calisto MT"))  
 self.invalidText.SetForegroundColour(wx.RED)  
  
 *# place invalid text in right position in sizer* sizer = self.GetSizer()  
 sizer.Insert(sizer.GetItemCount(), self.invalidText, 0, wx.ALIGN\_TOP | wx.CENTER, 10)  
  
 *# Disable interactive elements* self.codeField.Disable()  
 self.can\_press = False  
 self.connect.SetBitmap(wx.Bitmap("connect\_off.png"))  
 self.Layout()  
  
 timer = wx.Timer(self)  
 self.Bind(wx.EVT\_TIMER, self.on\_timer, timer)  
 *# start timer for certain time gotten* timer.Start(time, oneShot=True)  
  
 def on\_timer(self, event):  
 *"""  
 Function called when timer ends and erases invalid text currently seen  
 :param event: event for time  
 """  
 # Remove the invalid message if still exists and enable button and textctrl back* if self.invalidText is not None:  
 self.invalidText.Destroy()  
 self.Layout()  
 self.codeField.Enable()  
 self.can\_press = True  
 self.connect.SetBitmap(wx.Bitmap("connect.png"))  
  
  
class ConnectingPanel(wx.Panel):  
 def \_\_init\_\_(self, parent, frame):  
 *"""  
 Connecting to session panel  
 :param parent: Parent panel  
 :param frame: frame parent  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, style=wx.SIMPLE\_BORDER, size=(1920, 1080))  
 self.frame = frame  
 self.parent = parent  
 self.SetBackgroundColour("#fdf0d0")  
  
 self.sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# Initialize title on panel* title = wx.StaticText(self, -1, label="UELP")  
 titlefont = wx.Font(68, wx.DEFAULT, wx.NORMAL, wx.NORMAL, False, "Algerian")  
 title.SetForegroundColour("#aa7c57")  
 title.SetFont(titlefont)  
  
 *# Initialize connecting to session text on panel* self.connecting\_text = wx.StaticText(self, -1, label="Connecting to session")  
 connecting\_font = wx.Font(90, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 self.connecting\_text.SetForegroundColour("#3f4043")  
 self.connecting\_text.SetFont(connecting\_font)  
  
 *# Initialize disconnect from session instructions on panel* disconnect\_instruction = wx.StaticText(self, -1,  
 label="To disconnect from session press: Ctrl + Shift + D")  
 disconnect\_font = wx.Font(25, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, False, "Garamond")  
 disconnect\_instruction.SetForegroundColour("#880808")  
 disconnect\_instruction.SetFont(disconnect\_font)  
  
 *# Add elements to sizer* self.sizer.Add(title, 0, wx.ALIGN\_CENTER\_HORIZONTAL | wx.TOP, 5)  
 self.sizer.AddStretchSpacer()  
 self.sizer.Add(self.connecting\_text, 0, wx.ALIGN\_CENTER\_HORIZONTAL | wx.TOP, 5)  
 self.sizer.AddStretchSpacer()  
 self.sizer.Add(disconnect\_instruction, 0, wx.ALIGN\_CENTER\_HORIZONTAL | wx.BOTTOM, 20)  
  
 *# Add logo at the bottom right of panel* wx.StaticBitmap(self, -1, wx.Bitmap("logo.png"), pos=(1680, 920))  
  
 *# arrange the screen* self.SetSizer(self.sizer)  
 self.Layout()  
 self.Hide()  
  
 *# Initialize timer for dot animation* self.timer = wx.Timer(self)  
 self.Bind(wx.EVT\_TIMER, self.on\_timer)  
 self.dots\_counter = 0  
  
 def on\_timer(self, event):  
 *"""  
 Function called when timer ends, add dot to end of connecting text dynamically  
 :param event: event of timer ending  
 """* text = self.connecting\_text.GetLabel()  
  
 *# if text already has three dots start new set of dots and update the dot set counter otherwise add dot to text* if text.endswith("..."):  
 self.connecting\_text.SetLabel("Connecting to session")  
 self.dots\_counter += 1  
 else:  
 self.connecting\_text.SetLabel(text + ".")  
  
 *# Stop after reaching 3 sets of 3 dots and close connecting frame as user is connecting to session* if self.dots\_counter == 3:  
 self.timer.Stop()  
 self.parent.frame.Close()  
  
 def start\_dots\_animation(self):  
 *"""  
 Function called to start dot animation (new dot every half second)  
 :return:  
 """* self.timer.Start(500)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = wx.App()  
 test = MyFrame("A")  
 app.MainLoop()

def unpackData(data: str) -> str and list:  
 *"""  
 function gets data and unpacks it based on known opcodes and params protocols  
 :param data: data to unpack  
 :return: the opcode of the data and a list of all it's params  
 """* opcode = data[:2]  
 params = []  
 if opcode == "01":  
 params.append(data[2:6])  
 params.append(data[6:10])  
 params.append(data[10:14])  
 params.append(data[14:18])  
 params.append(data[18:])  
 elif opcode == "02":  
 params.append(data[2:])  
  
 return opcode, params  
  
  
def pack\_key\_click(unicode\_key: str) -> str:  
 *"""  
 function packs unicode key based on the protocol  
 :param unicode\_key: unicode\_key clicked  
 :return: packed data based on protocol  
 """* return "01" + unicode\_key.zfill(7)  
  
  
def pack\_key\_release(unicode\_key: str) -> str:  
 *"""  
 function packs unicode key based on the protocol  
 :param unicode\_key: unicode\_key clicked  
 :return: packed data based on protocol  
 """* return "02" + unicode\_key.zfill(7)

import queue  
import threading  
  
from Monitor\_Keyboard import Keyboard\_monitor  
from Monitor\_Mouse import Mouse\_monitor  
from Servercomm import Server\_comm  
  
  
def check(check\_queue, listener):  
 *"""  
 Function checks if Assistance Seeker client disconnected from server, if yes stop listener for keyboard/mouse  
 :param check\_queue: queue for comm between server object and logic  
 :param listener: listener object of keyboard/mouse to listen to computer  
 """* while True:  
 data, ip = check\_queue.get()  
 if data == "disconnect":  
 listener.stop\_listening()  
 break  
  
  
def main\_Helper(otherIP, port, close\_queue):  
 *"""  
 Main helper function, run listener for mouse or keyboard based on server port and start thread  
 for checking disconnect from server  
 :param otherIP: ip of Assistance Seeker client that needs to connect to server   
 :param port: port Helper server runs on  
 :param close\_queue: multiprocessing queue for Keyboard listener, to send close session request to main client to  
 terminate all running proccesses  
 """* check\_queue = queue.Queue()  
 *# create new server and monitor mouse or keyboard based on port gotten  
 # start thread for checking disconnect client from server* server = Server\_comm(check\_queue, port, otherIP)  
 if port == 2001:  
 mouse = Mouse\_monitor(server, otherIP)  
 threading.Thread(target=check, args=(check\_queue, mouse,)).start()  
 elif port == 2002:  
 keyboard = Keyboard\_monitor(server, otherIP, close\_queue)  
 threading.Thread(target=check, args=(check\_queue, keyboard,)).start()

import queue  
import zlib  
  
import pygame  
from PIL import Image  
  
from Servercomm import Server\_comm  
  
*# global current PIL image on screen*currScreen = None  
  
  
def build\_Screen(top: int, left: int, right: int, bottom: int, img\_bytes: bytes):  
 *"""  
 Function gets the location and bytes of the image to paste and pastes it on current screen  
 :param top: top value of Image  
 :param left: left value of Image  
 :param right: right value of Image  
 :param bottom: bottom value of Image  
 :param img\_bytes: bytes of Image  
 """  
 # location to paste Image* paste\_loc = (top, left)  
  
 *# Create a PIL Image object from the bytes of the image bytes* diff\_img = Image.frombytes('RGB', (bottom - top, right - left), img\_bytes)  
  
 *# Paste the diff image on the current screen* currScreen.paste(diff\_img, paste\_loc)  
  
  
def main\_Helper\_screen(otherIP):  
 *"""  
 Main Helper screen, get images from Assistance Seeker client and paste  
 the images based on if are difference or full Images accordingly  
 :param otherIP: ip of Assistance Seeker client getting images from  
 """* recv\_q = queue.Queue()  
 port = 2003  
 global currScreen  
 *# create new server that only gets messages from otherIP with port and recv\_q gotten* server = Server\_comm(recv\_q, port, otherIP)  
  
 *# create pygame screen and save width and height of screen* pygame.init()  
 screen\_info = pygame.display.Info()  
 screen\_width, screen\_height = screen\_info.current\_w, screen\_info.current\_h  
  
 *# wait until gotten first image from Assistance seeker to create pygame screen* while True:  
 if not recv\_q.empty():  
 screen = pygame.display.set\_mode((screen\_width, screen\_height), pygame.FULLSCREEN)  
 break  
  
 while True:  
 *# get data from client* if not recv\_q.empty():  
 data, ip = recv\_q.get()  
 if data == "disconnect":  
 pygame.quit()  
 break  
  
 *# if amount of params in data is 5, we got a part image, take out params decompress and  
 # build\_screen based on imageData gotten* if len(data) == 5 and currScreen:  
 top = data[0]  
 left = data[1]  
 bottom = data[2]  
 right = data[3]  
 diff\_bytes = data[4]  
 diff\_bytes = zlib.decompress(diff\_bytes)  
  
 top, left, right, bottom = int(top), int(left), int(right), int(bottom)  
  
 *# call \_build\_screen with all params of image to paste this image on the current screen* build\_Screen(top, left, right, bottom, bytes(diff\_bytes))  
 *# if only one param we got a full Image, decompress and build it as current screen* elif len(data) == 1:  
 newScreen\_bytes = data[0]  
 newScreen\_bytes = zlib.decompress(newScreen\_bytes)  
 *# Create a PIL Image object from the bytes of the image bytes* currScreen = Image.frombytes('RGB', (screen\_width, screen\_height),  
 bytes(newScreen\_bytes))  
  
 *# if there is currently a valid PIL image object as current screen show it to user using pygame* if currScreen is not None:  
 *# transform PIL image object to pygame image* currScreen\_bytes = currScreen.tobytes()  
 pygame\_currScreen = pygame.image.frombuffer(currScreen\_bytes,  
 (screen\_width, screen\_height), "RGB")  
 *# show image* screen.blit(pygame\_currScreen, (0, 0))  
 pygame.display.flip()  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 break

import time  
  
import keyboard as Keyboard  
from pynput import keyboard  
from pynput.keyboard import Key  
  
import Helper\_protocol  
  
  
*# class to monitor keyboard*class Keyboard\_monitor:  
 def \_\_init\_\_(self, server, clientIP: str, close\_queue):  
 *"""  
 Builder function creates new Keyboard\_monitor object with the vars gotten and calls function \_monitor\_keyboard  
 :param server: server to send keyboard data through  
 :param clientIP: client ip to send keyboard data too  
 """* self.server = server  
 self.clientIP = clientIP  
 *# values for keys who do not have values in pynput* self.special\_keys\_mapping = {Key.ctrl\_l: 1114112,  
 Key.ctrl\_r: 1114113,  
 Key.shift\_l: 1114114,  
 Key.shift\_r: 1114115,  
 Key.alt\_l: 1114116,  
 Key.alt\_r: 1114117,  
 Key.cmd: 1114118,  
 Key.cmd\_l: 1114119,  
 Key.cmd\_r: 1114120,  
 Key.enter: 1114121,  
 Key.backspace: 1114122,  
 Key.tab: 1114123,  
 Key.space: 1114124,  
 Key.delete: 1114125,  
 Key.esc: 1114126,  
 Key.f1: 1114127,  
 Key.f2: 1114128,  
 Key.f3: 1114129,  
 Key.f4: 1114130,  
 Key.f5: 1114131,  
 Key.f6: 1114132,  
 Key.f7: 1114133,  
 Key.f8: 1114134,  
 Key.f9: 1114135,  
 Key.f10: 1114136,  
 Key.f11: 1114137,  
 Key.f12: 1114138,  
 Key.home: 1114139,  
 Key.end: 1114140,  
 Key.page\_up: 1114141,  
 Key.page\_down: 1114142,  
 Key.insert: 1114143,  
 Key.menu: 1114144,  
 Key.caps\_lock: 1114145,  
 Key.num\_lock: 1114146,  
 Key.scroll\_lock: 1114147,  
 Key.pause: 1114148,  
 Key.print\_screen: 1114149,  
 Key.shift: 1114150,  
 Key.ctrl: 1114151,  
 Key.alt: 1114152,  
 Key.up: 1114153,  
 Key.down: 1114154,  
 Key.left: 1114155,  
 Key.right: 1114156}  
 self.close\_queue = close\_queue  
 self.listener = None  
 self.\_monitor\_keyboard()  
  
 def pressed\_end(self):  
 *"""  
 Function is called when combination Shift + Ctrl + d is pressed, send data to client based on protocol and  
 release Shift Ctrl and d  
 """  
 # release shift ctrl and d* self.server.send(self.clientIP, Helper\_protocol.pack\_key\_release("9999999"))  
 self.stop\_listening()  
  
 *# wait 0.1 seconds for client to recieve end msg then place close in close\_queue to let client know  
 # to terminate all proccesses* time.sleep(0.1)  
 self.close\_queue.put("close")  
  
 def \_monitor\_keyboard(self):  
 *"""  
 Function monitors and listens to keyboard clicks and releases and calls functions accordingly  
 """* self.listener = keyboard.Listener(on\_press=self.\_on\_press, on\_release=self.\_on\_release)  
 *# add hotkey for end session combination and function for it* Keyboard.add\_hotkey("ctrl+shift+d", self.pressed\_end)  
 self.listener.start()  
  
 def stop\_listening(self):  
 *"""  
 Stops the keyboard listener.  
 """* if self.listener.is\_alive():  
 self.listener.stop()  
  
 def \_on\_press(self, key):  
 *"""  
 Function is called when key is pressed, and checks if key is a special key or not and sends value of  
 key based on protocol and special keys mapping  
 :param key: key pressed on keyboard  
 """* if key in self.special\_keys\_mapping.keys():  
 msg = str(self.special\_keys\_mapping[key])  
 else:  
 msg = str(ord(key.char))  
 msg = Helper\_protocol.pack\_key\_click(msg)  
  
 self.server.send(self.clientIP, msg)  
  
 def \_on\_release(self, key):  
 *"""  
 Function is called when key is released and checks if key is in special keys, sends the value of key based on  
 protocol and special keys mapping  
 :param key: key released on keyboard  
 """* if key in self.special\_keys\_mapping.keys():  
 msg = str(self.special\_keys\_mapping[key])  
 else:  
 msg = str(ord(key.char))  
 msg = Helper\_protocol.pack\_key\_release(msg)  
 self.server.send(self.clientIP, msg)

import queue  
  
import pynput  
  
from Servercomm import Server\_comm  
  
  
*# class to monitor mouse*class Mouse\_monitor:  
 def \_\_init\_\_(self, server, clientIP: str):  
 *"""  
 Builder function creates new Mouse\_monitor object to monitor mouse functions  
 :param server: server to send through data about mouse  
 :param clientIP: ip of client to send mouse data too  
 """* self.server = server  
 self.clientIP = clientIP  
 self.listener = None  
  
 self.\_monitor\_mouse()  
  
 def \_monitor\_mouse(self):  
 *"""  
 Function creates a new listener for the mouse to listen for scroll clicks and movement  
 """* self.listener = pynput.mouse.Listener(on\_move=self.\_on\_move, on\_click=self.\_on\_click,  
 on\_scroll=self.\_on\_scroll)  
 self.listener.start()  
  
 def stop\_listening(self):  
 *"""  
 Stops the mouse listener.  
 """* if self.listener:  
 self.listener.stop()  
  
 def \_on\_move(self, x: int, y: int):  
 *"""  
 Function called when listener detects new pos of mouse and sends the pos to client by protocol  
 :param x: x loc of mouse  
 :param y: y loc of mouse  
 """* self.server.send(self.clientIP, str(x).zfill(4) + str(y).zfill(4) + "4")  
  
 def \_on\_click(self, x: int, y: int, button, pressed):  
 *"""  
 Function checks if mouse clicked and sends gotten data to client by protocol  
 :param x: x pos of mouse click  
 :param y: y pos of mouse click  
 :param button: button to check mouse type click  
 :param pressed: check pressed or released  
 """* msg = str(x).zfill(4) + str(y).zfill(4)  
 if button == pynput.mouse.Button.left:  
 *# check if button was pressed or released and add to protocol accordingly* if pressed:  
 msg += "0"  
 else:  
 msg += "5"  
 elif button == pynput.mouse.Button.right:  
 *# check if button was pressed or released and add to protocol accordingly* if pressed:  
 msg += "1"  
 else:  
 msg += "6"  
  
 *# send data to client by protocol* self.server.send(self.clientIP, msg)  
  
 def \_on\_scroll(self, x: int, y: int, dx: int, dy: int):  
 *"""  
 Function checks and listens to mouse scroll wheel and sends to client data if mouse scrolled by protocol  
 :param x: x pos of mouse scroll  
 :param y: y pos of mouse scroll  
 :param dx: added x for mouse scroll  
 :param dy: added y for mouse scroll  
 """* msg = str(x).zfill(4) + str(y).zfill(4)  
 if dy > 0:  
 msg += "2"  
 else:  
 msg += "3"  
  
 *# send data to client by protocol* self.server.send(self.clientIP, msg)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 m = Mouse\_monitor(Server\_comm(queue.Queue(), 2000, "0.0.0.0"), "127.0.0.1")

import queue  
import threading  
  
import keyboard as Keyboard  
from pynput import keyboard  
from pynput.keyboard import Key  
  
import AssistanceSeeker\_protocol as protocol  
from clientComm import Client\_comm  
  
  
def check\_close(close\_queue, client):  
 *"""  
 Function waits until detected close session combo closes client and puts close in close queue  
 :param close\_queue: multiprocessing queue between main client and keyboard process to  
 let main client know when to close session  
 :param client: client object to comm with server  
 """* while True:  
 if Keyboard.is\_pressed("shift+ctrl+d"):  
 close\_queue.put("close")  
 client.close()  
 break  
  
  
def main\_AS\_keyboard(otherIP, close\_queue):  
 *"""  
 Function controls keyboard click and releases gotten from Helper, unpack data and register  
 key click on computer keyboard  
 :param otherIP: ip of server of Keyboard helper   
 :param close\_queue: multiprocessing queue between main client and keyboard process to  
 let main client know when to close session  
 """* port = 2002  
 recv\_q = queue.Queue()  
 client = Client\_comm(otherIP, port, recv\_q)  
  
 *# keyboard controller object* keyboard\_cont = keyboard.Controller()  
  
 *# values for keys who do not have values in pynput* special\_keys\_mapping = {  
 1114112: Key.ctrl\_l,  
 1114113: Key.ctrl\_r,  
 1114114: Key.shift\_l,  
 1114115: Key.shift\_r,  
 1114116: Key.alt\_l,  
 1114117: Key.alt\_r,  
 1114118: Key.cmd,  
 1114119: Key.cmd\_l,  
 1114120: Key.cmd\_r,  
 1114121: Key.enter,  
 1114122: Key.backspace,  
 1114123: Key.tab,  
 1114124: Key.space,  
 1114125: Key.delete,  
 1114126: Key.esc,  
 1114127: Key.f1,  
 1114128: Key.f2,  
 1114129: Key.f3,  
 1114130: Key.f4,  
 1114131: Key.f5,  
 1114132: Key.f6,  
 1114133: Key.f7,  
 1114134: Key.f8,  
 1114135: Key.f9,  
 1114136: Key.f10,  
 1114137: Key.f11,  
 1114138: Key.f12,  
 1114139: Key.home,  
 1114140: Key.end,  
 1114141: Key.page\_up,  
 1114142: Key.page\_down,  
 1114143: Key.insert,  
 1114144: Key.menu,  
 1114145: Key.caps\_lock,  
 1114146: Key.num\_lock,  
 1114147: Key.scroll\_lock,  
 1114148: Key.pause,  
 1114149: Key.print\_screen,  
 1114150: Key.shift,  
 1114151: Key.ctrl,  
 1114152: Key.alt,  
 1114153: Key.up,  
 1114154: Key.down,  
 1114155: Key.left,  
 1114156: Key.right}  
  
 *# start thread for checking close session combo and comm with main client* closed = threading.Thread(target=check\_close, args=(close\_queue, client,))  
 closed.start()  
  
 while True:  
 data = recv\_q.get()  
 if data != "close":  
 opcode, key\_val = protocol.unpackData(data)  
 key\_val = int(key\_val)  
 *# gotten end session opcode from Helper, release end session combo and end loop* if key\_val == 9999999:  
 keyboard\_cont.release(Key.shift)  
 keyboard\_cont.release(Key.ctrl)  
 keyboard\_cont.release("d")  
 print("released end session combo")  
 break  
 *# check if value gotten is registered in special keys and translate to key accordingly* if key\_val in special\_keys\_mapping.keys():  
 key = special\_keys\_mapping[key\_val]  
 else:  
 *# add value for keys gotten between 0 and 27 as  
 # these keys are (Ctrl + key) and are not registered correctly in pynput* if 0 < key\_val < 27:  
 key\_val += 96  
 key = chr(key\_val)  
  
 *# check if key is released or clicked and change computer keyboard accordingly* if opcode == "01":  
 keyboard\_cont.press(key)  
 print("pressed " + key)  
 elif opcode == "02":  
 keyboard\_cont.release(key)  
 print("released " + key)  
 else:  
 break

import queue  
  
from pynput import mouse  
  
from clientComm import Client\_comm  
  
  
def main\_AS\_mouse(otherIP):  
 *"""  
 main loop for getting mouse data from Helper mouse, get data unpack it and change mouse control accordingly  
 :param otherIP: ip of Helper server getting mouse data from  
 """* port = 2001  
 recv\_q = queue.Queue()  
 client = Client\_comm(otherIP, port, recv\_q)  
  
 *# Computer mouse control object* mouse\_cont = mouse.Controller()  
  
 *# get mouse pos and other functions from queue* while True:  
 data = recv\_q.get()  
 if data != "close":  
 *# unpack mouse pos and type of function mouse has to do* x = data[:4]  
 y = data[4:-1]  
 type\_mov = data[-1]  
  
 *# move mouse and do func based on protocol* mouse\_cont.position = (int(x), int(y))  
 *# left click* if type\_mov == "0":  
 mouse\_cont.press(mouse.Button.left)  
 *# right click* elif type\_mov == "1":  
 mouse\_cont.press(mouse.Button.right)  
 *# scroll up* elif type\_mov == "2":  
 mouse\_cont.scroll(0, 1)  
 *# scroll down* elif type\_mov == "3":  
 mouse\_cont.scroll(0, -1)  
 *# left release* elif type\_mov == "5":  
 mouse\_cont.release(mouse.Button.left)  
 *# right release* elif type\_mov == "6":  
 mouse\_cont.release(mouse.Button.right)

def unpackData(data: str) -> tuple:  
 *"""  
 function gets data and unpacks it based on known opcodes and params protocols  
 :param data: data to unpack  
 :return: the opcode of the data and the param of the opcode  
 """* return data[:2], data[2:]  
  
  
def pack\_part\_image(top: str, left: str, bottom: str, right: str, img\_data\_size: str) -> str:  
 *"""  
 function packs part image header  
 :param left: left value of image place  
 :param top: top value of image place  
 :param right: right value of image place  
 :param bottom: bottom value of image place  
 :param img\_data\_size: length of data of img  
 :return: packed string of part image header  
 """* return "01" + top.zfill(4) + left.zfill(4) + bottom.zfill(4) + right.zfill(4) + img\_data\_size.zfill(10)  
  
  
def pack\_full\_image(img\_data\_size: str) -> str:  
 *"""  
 function packs full image header  
 :param img\_data\_size: length of data of img  
 :return: packed string of full image header  
 """* return "02" + img\_data\_size.zfill(10)

import queue  
import zlib  
  
import pygame  
from PIL import ImageChops, ImageGrab  
  
import AssistanceSeeker\_protocol  
from clientComm import Client\_comm  
  
  
def getChanged(newScreen: ImageGrab, currScreen: ImageGrab):  
 *"""  
 Function checks if there is a difference between two images, if there is it returns the difference Image  
 (Image that represents the differnce between the two Images) and the bounding box of the difference Image  
 :param newScreen: the ImageGrab image of the new screen that might differ from the currScreen  
 :param currScreen: the ImageGrab image of the current screen  
 :return: if the Images don't differ return None otherwise return the diff Image and the bounding box of this Image  
 """  
 # Check if the images are equal* diff = ImageChops.difference(currScreen, newScreen)  
 bbox = diff.getbbox()  
  
 *# Images are identical* if bbox is None:  
 return None, None  
  
 *# Get the region of the difference* diff\_region = newScreen.crop(bbox)  
  
 return diff\_region, bbox  
  
  
def getChangedPrecentage(bbox, total\_pixels: int) -> float:  
 *"""  
 Function checks how much percentage Image changed since the last Image based on bounding box of diffImage  
 :param bbox: The bounding box of the difference Image on the screen  
 :param total\_pixels: the total amount of pixels on screen  
 :return: percentage representing how much the Image has changed based on the difference Image  
 """  
 # get loc of diff Image* top, left, bottom, right = bbox  
  
 *# calculate the percentage of difference based on the pixel amount of the difference in total pixels of screen* diff\_pixels = (bottom - top) \* (right - left)  
 percentage\_difference = (diff\_pixels / total\_pixels) \* 100  
  
 return percentage\_difference  
  
  
def main\_AS\_screen(otherIP):  
 *"""  
 Main Assistance Seeker screen, get new screenshot of screen check if there is difference and  
 send the difference or the full new screenshot accordingly  
 :param otherIP: ip of Helper server to send images too  
 """* pygame.init()  
 screen\_info = pygame.display.Info()  
 screen\_width, screen\_height = screen\_info.current\_w, screen\_info.current\_h  
 recv\_q = queue.Queue()  
 port = 2003  
  
 client = Client\_comm(otherIP, port, recv\_q)  
  
 *# get first screenshot and compress the image* currScreen = ImageGrab.grab()  
 currScreen\_bytes = currScreen.tobytes()  
 currScreen\_bytes = zlib.compress(currScreen\_bytes)  
  
 *# wait for client to finish key exchange then send the first image* while not client.exchangeStatus():  
 pass  
  
 *# send first screenshot to Helper based on protocol* client.sendImage(AssistanceSeeker\_protocol.pack\_full\_image(str(len(currScreen\_bytes))), currScreen\_bytes)  
  
 while True:  
 *# get new screenshot* newScreen = ImageGrab.grab()  
 diffImage, diff\_bbox = getChanged(newScreen, currScreen)  
 if diffImage is not None:  
 changed\_precentage = getChangedPrecentage(diff\_bbox, screen\_height \* screen\_width)  
 *# if Image is more then 75 percent different, send the entire Image and not the difference* if changed\_precentage >= 75:  
 newScreen\_bytes = newScreen.tobytes()  
 newScreen\_bytes = zlib.compress(newScreen\_bytes)  
 *# send full screenshot to Helper based on protocol* client.sendImage(AssistanceSeeker\_protocol.pack\_full\_image(str(len(newScreen\_bytes))),  
 newScreen\_bytes)  
  
 else:  
 *# get the bounding box values of the difference Image* top, left, bottom, right = diff\_bbox  
  
 diffBytes = diffImage.tobytes()  
 diffBytes = zlib.compress(diffBytes)  
 *# build difference Image header based on protocol* msg = AssistanceSeeker\_protocol.pack\_part\_image(str(top), str(left), str(bottom), str(right),  
 str(len(diffBytes)))  
  
 *# send difference Image and Image header to Helper* client.sendImage(msg, diffBytes)  
  
 *# update currScreen as newScreen* currScreen = newScreen