

**מגיש: אופיר קרזבום**

**ת"ז: 215427204**

**מורה: מרי גבע**

**בית הספר: כרמים בנימינה**

**תאריך: 18.5.2024**

Contents

[פרק א' – reflection – ייזום 3](#_Toc166938844)

[פרק ב' - reflection - אפיון 6](#_Toc166938845)

[פרק ג' - reflection- מסמך ניתוח 11](#_Toc166938846)

[פרק ד' - reflection - העיצוב 26](#_Toc166938847)

[פרק ה' - reflection - הקוד 62](#_Toc166938848)

[פרק ו' - reflection – בדיקות ('קופסא לבנה') 68](#_Toc166938849)

[פרק ז' - reflection – מדריך למשתמש 70](#_Toc166938850)

[פרק ח' - reflection – רפלקציה 72](#_Toc166938851)

[פרק ט' - reflection – ביבליוגרפיה 73](#_Toc166938852)

[קוד הפרויקט 74](#_Toc166938853)

# פרק א' – reflection – ייזום

**1. תיאור ראשוני:**

פעמים רבות כאשר תלמידים מגיעים למעבדה נוצרת בעיה בכך שמחשב שתלמיד עבד עליו בשיעור הקודם ונשמרו בו קבצים הכרחיים , נלקח ע"י תלמיד אחר. בשביל לפתור זאת נשתמש במערכת שמירת קבצים מבוזרת. מערכת זאת תשמור כל קובץ שתלמיד שינה או יצר ובכך בכל פעם שהתלמיד יתחבר ממחשב אחר כאשר המחשב שעבד עליו בשיעור הקודם פתוח יהיה לו את האפשרות לגשת ולשנות את הקבצים.

בחרתי בפרויקט זה משום שרציתי פרויקט שיתעסק בעבודה עם הרשת מכיוון שנושא זה מאוד מעניין אותי, בנוסף נתקלתי בבעיה זו של שמירת הקבצים במחשב אחד שלא יכולתי לגשת אליהם מכיוון שתלמיד אחר התיישב במקום של המחשב שעבדתי עליו בשיעור הקודם.

האתגר המרכזי של הפרויקט יהיה לגשת אל הקבצים ללא שרת. תחילה אבנה את הפרויקט בצורה של שרת שיודיע ללקוח (המיוצג כמחשב) איפה נמצאים הקבצים שלו, לשם הלקוח יצטרך לגשת ולקחת אותם. כדי לגשת אל הקבצים ללא שרת הלקוח יצטרך לשלוח broadcast לכל המחשבים ברשת ולשאול אצל מי נמצא הקבצים שלו. לבנות את בקשת הbroadcast ושליחת הקבצים יהיו האתגר המרכזי בפרויקט. בנוסף שמירת הקבצים בצורה בטוחה שרק הלקוח המתאים יוכל לגשת אליהם היא אתגר נוסף.

**2. הגדרת הלקוח:**

הפרויקט מיועד למעבדות מחשבים באוניברסיטאות ובבתי ספר בהם אין מחשב אישי מוגדר לכל תלמיד.

**3. הגדרת יעדים/מטרות:**

יש לפרויקט 2 מטרות מרכזיות:

1. לשמור את הקבצים בצורה בטוחה, רק התלמיד הספציפי יוכל לגשת לקבצים שלו, אין אנשים אחרים שיקבלו הרשאה לגשת אל הקבצים של אותו תלמיד
2. התלמיד יוכל לגשת אל הקבצים שלו מכל מחשב במעבדת המחשבים כל עוד המחשבים שעבד עליהם בפעמים הקודמות דלוקים.

**4.** **בעיות, תועלות וחסכונות:**

הבעיה שנוצרה היא שתלמיד שעובד על מחשב מסוים בשיעור אחד אך בשיעור אחר נלקח לו המחשב, אין לתלמיד אפשרות לגשת אל הקבצים שעבד עליו בפעם הקודמת ובכך הוא יצטרך ליצור את הקבצים מחדש ובכך לבזבז המון זמן.

התועלת של הפרויקט היא עזרה לתלמידים לשמור את הקבצים שלהם בצורה יותר נוחה ללא שימוש באביזרים חיצוניים כגון דיסק און קי או אפילו במייל ובמתן אפשרות לתלמידים לגשת לקבצים שלהם ששמורים על מחשבים שונים ברחבי הרשת מכל מחשב אחר ברשת ללא צורך בהעלאתם לאחסון מרכזי.

פתרון קיים שעונה על אותו הצורך של המערכת הוא דיסק און קי. דיסק און קי גם כן מטרתו לשמור את הקבצים שעליו עבד התלמיד אך בשונה מהמערכת הדיסק און קי הוא כלי פיזי לכן אפשר לשכוח אותו ובנוסף צריך לשמור את הקבצים בצורה ידנית ברגע שהתלמיד מתנתק מהמחשב.

פתרון קיים נוסף שעונה על אותו הצורך של המערכת הוא תיקייה משותפת ברשת. בדומה למערכת תיקייה משותפת ברשת שומרת את הקבצים של המשתמש והתלמיד יוכל לגשת אל הקבצים מכל מקום במעבדה לעומת זאת הקבצים של התלמיד אינם יהיו מוגנים לכן כל אדם שמתחבר לרשת יוכל למחוק את הקבצים של התלמיד.

פתרון קיים נוסף שעונה על אותו הצורך של המערכת הוא מקום אחסון ברשת לדוגמא ענן, אבל פתרון זה מצריך את התלמיד בביצוע פעולות של העלאת והורדת קבצים לענן וכן ברישום לענן כלשהו.

הפתרון שלי לא מצריך מהמשתמש לשנות את הרגליו אלא דואג לאפשר לו לגשת לקבצים עליהם עבד במקומות שונים ברשת מכל מחשב אחר ברשת.

**5. מגבלות בהגדרת המערכת:**

הפרויקט לא מתעסק עם טכנולוגיה חדשה ובלתי מוכרת.

מכיוון שהרשתות של בתי הספר הם רשתות סגורות אין אפשרות לתלמיד לגשת אל הקבצים שלו מביתו לכן מערכת שמירת הקבצים תעבוד רק בבית הספר.

**6. תיחום הפרויקט:**

הפרויקט עובד במיוחד עם רשתות מכיוון שהעברת הקבצים ובקשת הקבצים תעבור בתוך רשת ספציפית. בנוסף הפרויקט יעבוד עם מערכת הפעלה ווינדוס משום שבפרויקט אנחנו עובדים עם העברת קבצים וכן יעשה שימוש ב thread-ים למיקבול תהליכים ועוד. הפרויקט יעבוד גם עם קריפטוגרפיה מכיוון שנצטרך להצפין את המידע שמועבר בתוך הרשת של הקבצים.

**7. קיימות כדור הארץ:**

כרגע הפתרון העיקרי לבעיית שמירת הקבצים הוא דיסק און קי. הייצור של דיסק און קי עולה כסף ומשאבים מזהמים לכן מערכת שמירת הקבצים עדיפה בהרבה ושומרת על כדור הארץ.

# פרק ב' - reflection - אפיון

**1. פרוט המערכת:**

פעמים רבות כאשר תלמידים מגיעים למעבדה נוצרת בעיה בכך שמחשב שתלמיד עבד עליו בשיעור הקודם ונשמרו בו קבצים הכרחיים , נלקח ע"י תלמיד אחר. בשביל לפתור זאת נשתמש במערכת שמירת קבצים מבוזרת. מערכת זאת תשמור כל קובץ שתלמיד שינה או יצר ובכך בכל פעם שהתלמיד יתחבר ממחשב אחר כאשר המחשב שעבד עליו בשיעור הקודם פתוח ומחובר לרשת יהיה לו את האפשרות לגשת ולשנות את הקבצים.

תחילה אבנה את הפרויקט בצורה של שרת שיודיע ללקוח (המיוצג כמחשב) איפה נמצאים הקבצים שלו, לשם הלקוח יצטרך לגשת ולקחת אותם. כדי לגשת אל הקבצים ללא שרת הלקוח יצטרך לשלוח broadcast לכל המחשבים ברשת ולשאול אצל מי נמצא הקבצים שלו.

**3. יכולות המשתמש:**

התחברות למערכת – כאשר משתמש פותח את המחשב יופיע לו 3 אפשרויות. או להתחבר למערכת עם שם משתמש וסיסמא, ליצור משתמש חדש או להיכנס כאורח (לבחור באופציה זו לא ייתן למשתמש את האפשרות להשתמש ביכולות הבאות).

שמירת קבצים - כמו בכל מחשב רגיל, המשתמש יוכל לשמור את הקבצים שהוא יוצר ולשמור אותם בדיסק D ובכך הקבצים שלו ישמרו לפעם הבאה שיתחבר למערכת.

שינוי קבצים שמורים - בדומה לשמירת הקבצים שהמשתמש יצר, המשתמש יוכל לשנות את תוכן, שמות הקבצים ששמר והשינויים ישמרו לפעם הבאה שיתחבר למערכת.

קבלת קבצים שמורים - כאשר משתמש מתחבר למערכת, יופיע למשתמש כל הקבצים אשר עבד עליהם בפעמים קודמות בכל המחשבים הפזורים ברשת.

**4. פירוט הבדיקות ('קופסא שחורה')**

| **מספר** | **שם הבדיקה (שם שיעיד על תוכן הבדיקה)** | **מה אמורה לבדוק** | **איך מתכננים לבדוק**  **(לתאר בפירוט את שלבי הבדיקה)** |
| --- | --- | --- | --- |
| 1 | יצירת קבצים | לאחר התחברות למערכת, האם כאשר יוצרים במחשב א' קובץ, הקובץ יימצא במראה הקבצים במחשב ב'. | מתחברים למערכת במחשב א', יוצרים קובץ. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם הקובץ נמצא במראה הקבצים במחשב ב' |
| 2 | מחיקת קבצים | לאחר התחברות למערכת, האם כאשר מוחקים במחשב א' קובץ, הקובץ לא יימצא במראה הקבצים במחשב ב'. | עושים זאת לאחר בדיקה מס' 1.  מתחברים למערכת במחשב א', מוחקים את הקובץ שיצרנו. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם הקובץ נעלם ממראה הקבצים. |
| 3 | שינוי שם קובץ | לאחר התחברות למערכת, האם כאשר משנים של קובץ במחשב א', שם הקובץ ישתנה במראה הקבצים במחשב ב'. | עושים זאת לאחר בדיקה מס' 1.  מתחברים למערכת במחשב א', משנים את שם הקובץ שיצרנו. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם שם הקובץ ישתנה במראה הקבצים. |
| 4 | שינוי תוכן קובץ | לאחר התחברות למערכת, האם כאשר משנים תוכן של קובץ במחשב א', תוכן הקובץ ישתנה במחשב ב'. | עושים זאת לאחר בדיקה מס' 1.  מתחברים למערכת במחשב א', כותבים בתוך הקובץ (txt) כמה מילים. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם מה שכתבנו בקובץ במחשב א' נמצא בקובץ במחשב ב'. |
| 5 | סנכרון פעולות | נבדוק האם כל הפעולות שכתבנו לעיל עובדות כאשר עושים אותם באותו זמן ושום דבר אינו מתבלבל | ניצור 2 קבצים במחשב א' נתחבר למחשב ב' ונבדוק כי שניהם נוצרו. לאחר מכן שוב נתחבר למחשב א' ניצור עוד קובץ, נשנה את שם הקובץ של אחד משני הקבצים ואת תוכן הקובץ ונמחק את הקובץ הנוסף. נתחבר למחשב ב' ונבדוק שכל הפעולות שעשינו התבצעו כראוי |
| 6 | זמני תגובה | האם בעבודה עם קבצים גדולים יש פגיעה בזמני התגובה של המערכת | נשמור 5 קבצים גדולים במחשב ב' דרך מחשב א' ונבדוק האם במחשבים ב' בו מחובר משתמש אחר ברשת יש עיכוב בפעולות רגילות |
| 7 | עומס | האם כשיש משתמשים רבים במערכת, הביצועים עדיין טובים | נחבר 4 משתמשים למערכת מ8 מחשבים שונים ונבצע כמה פעולות שמתוארות לעיל ונבדוק האם הפעולות מתבצעות כראוי |
| 8 | יציבות | האם המערכת נשארה יציבה אחרי התחברויות והתנתקויות של משתמשים | נחזור על פעולה מס' 7 ומדי פעם נחבר וננתק משתמשים לבדוק האם המערכת אינה קורסת |
| 9 | אבטחה | האם בעת העבודה על הקבצים ממחשב אחר המידע לא זולג החוצה | נפתח wireshark נחזור על פעולה 1 ונבדוק האם המידע שהועבר ברשת מוצפן או לא |
| 10 | העלאת קובץ | האם כאשר מעלים קובץ אפשרי לפתוח אותו דרך המערכת | מתחברים למחשב מס 1 מעלים קובץ מן המחשב, מתחברים למחשב מס 2 ובודקים אם הקובץ קיים והתוכן שלו נכון |
| 11 | הורדת קובץ | האם כאשר מורידים קובץ הוא מופיע במראה הקבצים של ווינדוס | מתחברים למחשב מס 1 מורידים קובץ ובודקים האם הוא נפתח במראה הקבצים של ווינדוס |
| 12 | העברת קובץ | האם כאשר מעבירים קובץ ממחשב אחד למחשב אחר הקובץ מופיע במחשב האחר | מתחברים למחשב מס 1 מעבירים קובץ מהמחשב למחשב מס 2. בודקים האם הקובץ נמחק מהמחשב. מתחברים למחשב מס 2 ובודקים האם הקובץ נמצא |
| 13 | שכפול קובץ | האם כאשר מעתיקים קובץ ממחשב אחד למחשב אחר הקובץ מופיע במחשב האחר | מתחברים למחשב מס 1 מעתיקים קובץ מהמחשב למחשב מס 2. מתחברים למחשב מס 2 ובודקים אם הקובץ נמצא |
| 14 | הרשמה למערכת | האם כאשר נרשמים למערכת עם שם משתמש שלא קיים מכניס למערכת הקבצים | נרשמים למערכת עם שם משתמש שלא קיים ובודקים אם נכנס למערכת קבצים. מריצים את המערכת מחדש ומנסים להירשם עם אותו משתמש ובודקים אם מוצגת הודעה ראויה שהמשתמש כבר קיים |
| 15 | התחברות למערכת | האם כאשר מתחברים למערכת עם שם משתמש קיים מכניס למערכת קבצים | מתחברים למערכת עם שם משתמש קיים ובודקים אם נכנס למערכת הקבצים. מריצים את המערכת מחדש ומנסים להיכנס עם שם משתמש שלא קיים ובודקים שמוצגת הודעה ראויה שהמשתמש לא קיים. |

**5. תכנון לוח זמנים לפרויקט**:

| **פעילות** | **זמן התחלה מתוכנן** | **זמן סיום מתוכנן** | **זמן התחלה בפועל** | **זמן סיום בפועל** | **הערות** |
| --- | --- | --- | --- | --- | --- |
| יזום | 6.11.2023 | 25.11.2023 | 6.11.2023 | 9.11.2023 |  |
| אפיון | 8.11.2023 | 1.12.2023 | 10.11.2023 | 16.11.2023 |  |
| ניתוח | 11.11.2023 | 15.12.2023 | 16.11.2023 | 3.12.2023 |  |
| עיצוב | 1.12.2023 | 15.1.2024 | 43.12.202 | 415.1.202 |  |
| פירוק הקוד לאבנים גדולות:  1) בניית שרת לקוח בסיסי, עבודה בסיסית עם קבצים  2) בניית בסיס נתונים  3) ממשק לקוח 1, עבודה עם מחשב אחד, מראה את הקבצים ששמר פעמים קודמות  4) עבודה עם 2 לקוחות. התחברות עם מחשב אחר ומראה קבצים שעבד פעמים קודמות  5) עבודה עם פעולות של קבצים  6) שרת מרובה לקוחות  7) הצפנה  8) החלפת שרת בבקשת ברודקאסט  9) יצירת מערכת קבצים חיה (live) שכאשר משנים\יוצרים\מוחקים קובץ תוצאת הפעולה תופיע ישירות בכל המחשבים בהם קבצים אלה פתוחים. | 1)415.1.202  2)20.1.2024  3) 24.1.2024  4) 1.2.2024  5) 7.2.2024  6) 12.2.2024  7) 15.2.2024  8) 17.2.2024  9) 25.2.2024 | 1) 20.1.2024  2) 24.1.2024  3) 1.2.2024  4) 7.2.2024  5) 12.2.2024  6) 15.2.2024  7) 17.2.2024  8) 25.2.2024  9) 1.3.2024 | 3)20.2.2024  1)415.1.202  4)25.2.2024  5)3.3.2024  6)10.3.2024  2)1.2.2024  1)21.1.2024  X  X | 4)25.2.2024  1)21.1.2024  5)3.3.2024  6)10.3.2024  7)5.5.2024  3)20.2.2024  2)1.2.2024  X  X | לא מתוכנן לעשות  מתוכנן לסיום בקרוב |
| גרסה ראשונית | 15.1.2024 | 1.3.2024 | 15.1.2024 | 5.5.2024 |  |
| מסמך בדיקות | 1.4.2024 | 10.4.2024 | 5.5.2024 | 12.5.2024 |  |
| מדריך למשתמש | 10.4.2024 | 1.5.2024 | 12.5.2024 | 14.5.2024 |  |
| גרסה סופית | 1.5.2024 | 10.5.2024 | 14.5.2024 | 18.5.2024 |  |
| סגירת תיק פרויקט | 1.5.2024 | 10.5.2024 | 14.5.2024 | 18.5.2024 |  |

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

**6. 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 | פרויקט הושלם | 18.5.2024 |
| אי הצלחת עבודה עם מערכת הקבצים של ווינדוס | לא אצליח להראות בדרך הנחוצה את השינויים בקבצים ועוד. | בינוני | * לנסות את העבודה עם הקבצים כמה שיותר מוקדם * במקרה הגרוע לבנות gui נפרד. | נבנה gui נפרד | 20.2.2024 |
| אי הצלחת מימוש הצפנה | כל המידע שמועבר ברשת לא יצליח לעבור מוצפן | קשה | * ניסיונות של כמה סוגי הצפנה אם ההצפנה הרצויה לא עובדת * לחקור יותר לעומק על סוגי הצפנה וההצפנה הנכונה ביותר למקרה הספציפי שלי | ההצפנה הושלמה | 1.2.2024 |
| ניהול קוד לא מאורגן | ניהול הקוד יהיה מבולגן ויכול לגרום לאי עמידה בזמנים | בינוני | * שמירה על קוד מאורגן מתחילת הפרויקט ועד סופו, הערות. * יצירת מחלקות ושמירה כללית מאורגנת של איפה נמצא הקוד | הקוד והמחלקות היו מסודרות במהלך כל הפרויקט | 5.5.2024 |
| מהימנות מראה הקבצים | לא ניתן ליצור מראה קבצים ללקוח שתואם למציאות | קשה | * התנהלות עם מראה הקבצים יהיה עדיפות ראשונית בפרויקט * נעשה במהלך כל הפרויקט בדיקות רבות שיבדקו את מהימנות מראה הקבצים | מראה הקבצים בנוי בצורה פשוטה וקלה להבנה | 5.5.2024 |
| תקשורת | לא ניתן לגשת לקבצים שלי ברחבי הרשת | קשה | * עבודה עם tcp כדי שלא יאבד מידע ברשת והמידע יעבור בצורה מהימנה * נעבוד עם wirkeshark במהלך הפרויקט כדי לבדוק את התקשורת | אפשר לגשת לקבצים בצורה קלה ופשוטה ללא סיבוכים | 5.5.2024 |

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

# פרק ג' - reflection- מסמך ניתוח

צד לקוח\שרת:

שם היכולת: **שליחת נתונים – צד לקוח\שרת**

מהות היכולת: שליחת נתונים כללית לפי פרוטוקול והצפנה

אוסף יכולות:

* בניית בקשה לפי פרוטוקול
* הצפנה
* שליחת נתונים לשרת\לקוח

רשימת אובייקטים: פרוטוקול, הצפנה, תקשורת

צד לקוח\שרת:

שם היכולת: **פירוק מידע שהתקבל – צד לקוח\שרת**

מהות היכולת: פירוק נתונים כללי לפי פרוטוקול והצפנה

אוסף יכולות:

* קבלת נתונים מלקוח\שרת
* פענוח
* פירוק לפי פרוטוקול

רשימת אובייקטים: פרוטוקול, פיענוח, תקשורת

צד לקוח:

שם היכולת: **הרשמה למערכת – צד לקוח**

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

אוסף יכולות:

* ממשק משתמש – מסך הרשמה
* קליטת נתונים
* בדיקת תקינות
* שליחת נתונים
* פירוק מידע שהתקבל
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש

צד שרת:

שם היכולת: **הרשמה למערכת – צד שרת­­**

מהות היכולת: רישום משתמש חדש במערכת

אוסף יכולות:

* פירוק מידע שהתקבל
* בדיקה מול בסיס הנתונים
* גיבוב סיסמא
* הוספה לבסיס נתונים
* שליחת נתונים

רשימת אובייקטים: בסיס נתונים

**צד לקוח:**

שם היכולת: **התחברות למערכת – צד לקוח**

מהות היכולת: התחברות משתמש קיים

אוסף יכולות:

* הרשמה למערכת – צד לקוח
* ממשק משתמש – מסך התחברות
* קליטת נתונים
* בדיקת תקינות
* שליחת נתונים
* פירוק מידע שהתקבל
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש

צד שרת:

שם היכולת: **התחברות למערכת – צד שרת**

מהות היכולת: התחברות משתמש קיים במערכת

אוסף יכולות:

* פירוק מידע שהתקבל
* גיבוב סיסמא
* בדיקה מול בסיס הנתונים
* שליחת נתונים

רשימת אובייקטים: בסיס נתונים

**צד לקוח:**

שם היכולת: **הצגת רשימת קבצים – צד לקוח**

מהות היכולת: להציג למשתמש את כל הקבצים שלו במערכת

אוסף יכולות:

* התחברות למערכת – צד לקוח
* פירוק מידע שהתקבל
* ממשק משתמש – מראה קבצים

רשימת אובייקטים: ממשק משתמש

**צד שרת:**

שם היכולת: **בניית רשימת קבצים – צד שרת**

מהות היכולת: בניית הודעה הכוללת את שמות קבצי המשתמש ומיקומם

אוסף יכולות:

* התחברות למערכת – צד שרת
* שליפת נתונים מבסיס הנתונים לפי שם משתמש
* בדיקת MAC קיים ברשת ועדכון רשימה בהתאם
* שליחת נתונים

רשימת אובייקטים: בסיס נתונים, MAC, תקשורת, רשימת קבצים

**צד לקוח :**

שם היכולת: **ניטור קובץ**

מהות היכולת: תפיסת אירועים בתיקייה המנוטרת ושליחתם לשרת

אוסף יכולות:

* האזנה
* תפיסת אירוע רלוונטי
* שליחת נתונים לשרת

**צד לקוח – כללי:**

שם היכולת: **יצירת קובץ – צד לקוח כללי**

מהות היכולת: קבלת קובץ

אוסף יכולות:

* הצגת רשימת קבצים – צד לקוח

**צד לקוח – בעל קובץ:**

שם היכולת: **יצירת קובץ – צד לקוח יוצר קובץ**

מהות היכולת: יצירת קובץ

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת מיקום קובץ – ממשק משתמש
* בחירת שם קובץ – ממשק משתמש
* יצירת קובץ
* שליחת נתונים

רשימת אובייקטים: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **יצירת קובץ – צד שרת**

מהות היכולת: יצירת קובץ והצגת הקובץ אצל כל הלקוחות

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* בדיקת תקינות שם ומיקום הקובץ
* פירוק מידע שהתקבל
* שליחת נתונים (לקוח כללי)

רשימת אובייקטים: קובץ

**צד לקוח - כללי:**

שם היכולת: **הצגת תוכן קובץ – צד לקוח כללי**

מהות היכולת: להוריד ולפתוח קובץ למשתמש

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת קובץ – ממשק משתמש
* שליחת נתונים
* פירוק מידע שהתקבל
* הצגת תוכן הקובץ

**צד לקוח – בעל קובץ:**

שם היכולת: **הצגת תוכן קובץ – צד לקוח בעל קובץ**

מהות היכולת: לשלוח קובץ לשרת

אוסף יכולות:

* פירוק מידע שהתקבל
* שליחת נתונים (קובץ)

רשימת אובייקטים לקוח: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **הצגת תוכן קובץ – צד שרת**

מהות היכולת: לשלוח למשתמש תוכן קובץ

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* שליחת נתונים (לבעל קובץ)
* פירוק מידע שהתקבל
* שליחת נתונים (למבקש הקובץ)

רשימת אובייקטים: קובץ

**צד לקוח - כללי:**

שם היכולת: **שינוי שם קובץ – צד לקוח כללי**

מהות היכולת: שינוי שם קובץ אצל כל המחשבים

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת קובץ – ממשק משתמש
* קליטת שם הקובץ ובדיקת תקינות
* שליחת נתונים
* הצגת רשימת קבצים – צד לקוח

**צד לקוח – בעל קובץ:**

שם היכולת: **הצגת תוכן קובץ – צד לקוח בעל קובץ**

מהות היכולת: שינוי שם קובץ שקובל

אוסף יכולות:

* פירוק מידע שהתקבל
* שינוי שם קובץ
* ניטור תיקייה

רשימת אובייקטים לקוח: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **שינוי שם קובץ – צד שרת**

מהות היכולת: שינוי שם קובץ אצל כל הלקוחות

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* שליחת נתונים (בקשה לשנות שם הקובץ אצל הלקוח בעל הקובץ)
* פירוק מידע שהתקבל (ניטור)
* בניית רשימת קבצים – צד שרת

רשימת אובייקטים: קובץ

**צד לקוח - כללי:**

שם היכולת: **מחיקת קובץ – צד לקוח - כללי**

מהות היכולת: מחיקת קובץ אצל כל המחשבים

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת קובץ – ממשק משתמש
* שליחת נתונים
* הצגת רשמית קבצים – צד לקוח

**צד לקוח – בעל קובץ:**

שם היכולת: **מחיקת קובץ – צד לקוח בעל קובץ**

מהות היכולת: מחיקת קובץ שקובל

אוסף יכולות:

* פירוק מידע שהתקבל
* מחיקת קובץ
* ניטור תיקייה

רשימת אובייקטים לקוח: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **מחיקת קובץ – צד שרת**

מהות היכולת: מחיקת קובץ אצל כל המשתמשים

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* שליחת נתונים (בעל קובץ)
* פירוק מידע שהתקבל
* בניית רשימת קבצים – צד שרת

רשימת אובייקטים: קובץ

**צד לקוח - כללי:**

שם היכולת: **שינוי תוכן קובץ – צד לקוח**

מהות היכולת: שינוי תוכן קובץ אצל מחשב מקורי

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* הצגת תוכן קובץ - לקוח
* כתיבת תוכן לשינוי
* שליחת נתונים

**צד לקוח – בעל קובץ:**

שם היכולת: **שינוי תוכן קובץ – צד לקוח בעל קובץ**

מהות היכולת: שינוי תוכן קובץ מקומי

אוסף יכולות:

* פירוק מידע שהתקבל
* שינוי תוכן קובץ
* שליחת נתונים (אישור)

רשימת אובייקטים לקוח: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **שינוי תוכן קובץ – צד שרת**

מהות היכולת: שינוי תוכן קובץ אצל כל הלקוחות

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* הצגת תוכן קובץ – צד שרת
* פירוק מידע שהתקבל
* שליחת נתונים (לבעל קובץ)
* פירוק מידע שהתקבל

רשימת אובייקטים: קובץ

**צד לקוח - כללי:**

שם היכולת: **העברת קובץ – צד לקוח כללי**

מהות היכולת: העברת קובץ ממיקום אחד לאחר

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת קובץ – ממשק משתמש
* בחירת מיקום חדש לקובץ – ממשק משתמש
* שליחת נתונים
* הצגת רשימת קבצים – צד לקוח

**צד לקוח – בעל קובץ:**

שם היכולת: **העברת קובץ – צד לקוח בעל קובץ**

מהות היכולת: העברת מיקום הקובץ

אוסף יכולות:

* פירוק מידע שהתקבל
* העברת מיקום
* ניטור תיקייה

רשימת אובייקטים לקוח: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **העברת קובץ – צד שרת**

מהות היכולת: העברת קובץ ממיקום אחד לאחר

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* בדיקת תקינות מקום שהועבר אליו
* שליחת נתונים (לבעל קובץ)
* פירוק מידע שהתקבל
* בניית רשימת קבצים – צד שרת

רשימת אובייקטים: אין

**צד לקוח - כללי:**

שם היכולת: **שכפול קובץ – צד לקוח כללי**

מהות היכולת: שכפול קובץ למקום מבוקש

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת קובץ – ממשק משתמש
* בחירת מיקום חדש לקובץ משוכפל – ממשק משתמש
* בחירת שם קובץ לקובץ משוכפל – ממשק משתמש
* שליחת נתונים
* הצגת רשימת קבצים צד לקוח

**צד לקוח – בעל קובץ:**

שם היכולת: **שכפול קובץ – צד לקוח בעל קובץ**

מהות היכולת: שכפול קובץ מבוקש

אוסף יכולות:

* פירוק מידע שהתקבל
* שכפול הקובץ למיקום מבוקש
* ניטור תיקייה

רשימת אובייקטים לקוח: ממשק משתמש, קובץ

**צד שרת:**

שם היכולת: **שכפול קובץ – צד שרת**

מהות היכולת: שכפול קובץ למקום מבוקש

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* בדיקת תקינות מקום שהועבר אליו
* בדיקת תקינות שם הקובץ
* שליחת נתונים (בעל קובץ)
* פירוק מידע שהתקבל
* בניית רשימת קבצים - צד שרת

רשימת אובייקטים: אין

**צד לקוח – כללי:**

שם היכולת: **יצירת תיקייה – צד לקוח כללי**

מהות היכולת: הצגת תיקייה שנוצרה

אוסף יכולות:

* הצגת רשימת קבצים – צד לקוח

**צד לקוח – בעל תיקייה:**

שם היכולת: **יצירת תיקייה – צד לקוח יוצר תיקיה**

מהות היכולת: יצירת תיקייה

אוסף יכולות:

* התחברות למערכת – צד לקוח
* בחירת מיקום התיקייה – ממשק משתמש
* בחירת שם תיקייה – ממשק משתמש
* יצירת תיקייה
* ניטור תיקייה

רשימת אובייקטים: ממשק משתמש

**צד שרת:**

שם היכולת: **יצירת תיקייה – צד שרת**

מהות היכולת: יצירת תיקייה

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* בדיקת תקינות שם ומיקום התיקייה
* פירוק מידע שהתקבל
* בניית רשימת קבצים – צד שרת

רשימת אובייקטים: אין

**צד לקוח - כללי:**

שם היכולת: **שינוי שם תיקייה – צד לקוח**

מהות היכולת: שינוי שם תיקייה אצל כל המחשבים

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת תיקייה – ממשק משתמש
* קליטת שם התיקייה ובדיקת תקינות
* שליחת נתונים
* הצגת רשימת קבצים – צד לקוח

רשימת אובייקטים: ממשק משתמש

**צד לקוח – בעל תיקייה:**

שם היכולת: **שינוי שם תיקייה – צד לקוח בעל תיקייה**

מהות היכולת: שינוי שם תיקייה

אוסף יכולות:

* פירוק מידע שהתקבל
* שינוי שם תיקייה
* ניטור תיקייה

**צד שרת:**

שם היכולת: **שינוי שם תיקייה – צד שרת**

מהות היכולת: שינוי שם תיקייה אצל כל הלקוחות

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* שליחת נתונים (בעל תיקייה)
* פירוק מידע שהתקבל
* בניית רשימת קבצים

רשימת אובייקטים: אין

**צד לקוח - כללי:**

שם היכולת: **מחיקת תיקייה – צד לקוח כללי**

מהות היכולת: מחיקת תיקייה אצל כל המחשבים

אוסף יכולות:

* התחברות למערכת – צד לקוח
* הצגת רשימת קבצים – צד לקוח
* בחירת תיקייה – ממשק משתמש
* שליחת נתונים
* הצגת רשימת קבצים – צד לקוח

**צד לקוח – בעל תיקייה:**

שם היכולת: **מחיקת תיקייה – צד לקוח בעל תיקייה**

מהות היכולת: שינוי שם תיקייה

אוסף יכולות:

* פירוק מידע שהתקבל
* מחיקת תיקייה
* ניטור תיקייה

רשימת אובייקטים: ממשק משתמש

**צד שרת:**

שם היכולת: **מחיקת תיקייה – צד שרת**

מהות היכולת: מחיקת תיקייה אצל כל המשתמשים

אוסף יכולות:

* התחברות למערכת – צד שרת
* בניית רשימת קבצים – צד שרת
* פירוק מידע שהתקבל
* שליחת נתונים (בעל תיקייה)
* פירוק מידע שהתקבל
* בניית רשמית קבצים – צד שרת

רשימת אובייקטים: אין

# פרק ד' - reflection - העיצוב

1. תיאור הארכיטקטורה של המערכת המוצעת

* החומרה בה הפרויקט משתמש היא מחשבי windows עם דיסק
* הלקוחות הם מחשבי windows עם דיסק, השרת הוא מחשב windows אשר מחובר לdatabase הלקוחות מתקשרים עם השרת אך גם מתקשרים עם הלקוחות האחרים בצורה של pear to pear. כל זה נמצא ברשת מקומית אחת.



1. תיאור הטכנולוגיה הרלוונטית

שפות התכנות שאשתמש בהם הם python wxpython,.וsql מערכת ההפעלה שנבחרה היא windows מכיוון שהמחשבים במעבדת המחשבים בבית ספר הם של windows, התקשורת מבוססת TCP מכיוון שאנחנו צריכים תקשורת אמינה בפרויקט שלי (אסור שיעבד מידע על קבצים).

תחומי עניין:

1. תקשורת – מימוש שרת מרובה לקוחות, תקשורת pear to pear, שימוש בסוקטים מעל פרוטוקול TCP
2. אבטחה – כל המידע עובר בתקשורת מוצפנת, בבסיס הנתונים מידע רגיש שמור כ hash.
3. מערכות הפעלה – שימוש בתהליכונים למיקבול, עבודה עם מערכת קבצים, שימוש בwinapi לטובת הניטור.
4. גרפיקה – שימוש בwxpython למימוש ממשק המשתמש
5. בסיס נתונים – שימוש בבסיס נתונים טבלאי sql
6. תיאור מודולים בהם נעשה שימוש:



סקירת כל המודולים המרכיבים את המערכת וקשרי הגומלין ביניהם:

**מודלים שאני מייבא:**

|  |  |
| --- | --- |
| מודול | לטובת |
| Socket | עבודה עם סוקטים בתקשורת |
| select | מימוש שרת מרובה לקוחות |
| Ast | לייצוג משתנים כמחרוזות בפרוטוקל |
| Ctypes | ליצירת תיקייה מוחבאת |
| Os | עבודה עם מחרוזות העוסקות במיקום של קבצים ותיקיות |
| Shutil | עבודה ישירה עם קבצים במחשב |
| Popen | בשביל פתיחת קבצים |
| Psutil | להתנהלות עם תהליכים |
| Queue | לשימוש בתור |
| Threading | לניהול תהליכונים |
| Sqlite3 | לניהול בסיס הנתונים |
| Pkcs1\_v1\_5 | להצפנה אסימטרית |
| rsa | ליצירת מפתחות אסימטריים |
| Base64 | להמרת בסיסים בהצפנה |
| Hashlib | למען גיבוב |
| Crypto | למען הצפנה |
| Sys | לסגירת התוכנית |
| time | לעיכוב התוכנית במקרים הנדרשים |
| getnode | לקבלת הmac |
| wx | גרפיקת הפרויקט |
| Pubsub | תקשורת בין גרפיקה ללוגיקה |
| Win32file | ניטור הקבצים |
| Functools | עבודה עם דברים בסיסיים בפייתון |

**מודלים שאני מייצר:**

| serverProtocol | | |
| --- | --- | --- |
| מחלקה עם אוסף פעולות לטיפול בפרוטוקול בצד שרת | | |
| פעולה | טענת כניסה | טענת יציאה |
| Unpack | data – הודעה שהתקבלה בתקשורת | מפרקת לפי הפרוטוקול ומחזירה tuple המכיל:  (opcode, [params]) |
| String\_to\_dict | Data – המחרוזת שמומרת למילון | מחזירה מילון המיוצג ע"י המחרוזת שהתקבלה |
| Dict\_to\_string | Data – המילון שמומר למחרוזת | מחזירה מחרוזת המיוצגת ע"י המילון שהתקבל |
| pack\_status\_register | Status – הצליח/לא הצליח (Boolean) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_login | Status – הצליח/לא הצליח (Boolean) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_ask\_file\_tree | Folder – השם של התיקייה שהשרת מבקש לקבל | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_send\_file\_tree | File\_tree - עץ הקבצים | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_create | Status – הצליח/לא הצליח (Boolean)  Path – המיקום של הקובץ  Typ – סוג הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_rename | Status – הצליח/לא הצליח (Boolean)  Path – המיקום של הקובץ  New\_name - שם הקובץ החדש | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_delete | Status – הצליח/לא הצליח (Boolean)  Location – מיקום הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_move | Status – הצליח/לא הצליח (Boolean)  Old\_path – המיקום הישן של הקובץ  New\_path – המיקום החדש של הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_clone | Status – הצליח/לא הצליח (Boolean)  Copy\_from – המיקום ממנו מעתיקים את הקובץ  Copy\_to – המיקום להעתיק אליו את הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_status\_mac | Status – הצליח/לא הצליח (Boolean) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_rename | Location – מיקום (str)  new\_name – שם חדש (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_delete | Location – מיקום (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_move | old\_location – מיקום ישן (str)  new\_location – מיקום חדש (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_clone | old\_location – מיקום ישן (str)  new\_location – מיקום חדש (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_create | Location – מיקום ליצור (str)  type – סוג יצירה (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_do\_remove | Folder – תיקייה להוריד ממראה הקבצים | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_public\_key | public\_key – מפתח ציבורי (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |

| userClientProtocol | | |
| --- | --- | --- |
| מחלקה עם אוסף פעולות לטיפול בפרוטוקול בצד לקוח משתמש | | |
| פעולה | טענת כניסה | טענת יציאה |
| unpack | data – הודעה שהתקבלה בתקשורת | מפרקת לפי הפרוטוקול ומחזירה tuple המכיל:  (opcode, [params]) |
| String\_to\_dict | Data – המחרוזת שמומרת למילון | מחזירה מילון המיוצג ע"י המחרוזת שהתקבלה |
| Pack\_register | Username – שם משתמש pass – סיסמה של המשתמש | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_sign\_in | Username – שם משתמש pass – סיסמה של המשתמש  Mac – הmac של המשתמש ברגע זה | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_create | Location – מיקום ליצור (str)  type – סוג יצירה (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_rename | Location – מיקום (str)  new\_name – שם חדש (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_delete | Location – מיקום (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_move | old\_location – מיקום ישן (str)  new\_location – מיקום חדש (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_clone | old\_location – מיקום ישן (str)  new\_location – מיקום חדש (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_do\_open\_file | Location – מיקום קובץ (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_change\_file | Location – מיקום קובץ (str) | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |

| generalClientProtocol | | |
| --- | --- | --- |
| מחלקה עם אוסף פעולות לטיפול בפרוטוקול בצד לקוח משתמש | | |
| פעולה | טענת כניסה | טענת יציאה |
| unpack | data – הודעה שהתקבלה בתקשורת | מפרקת לפי הפרוטוקול ומחזירה tuple המכיל:  (opcode, [params]) |
| Pack\_client\_type | Client\_type – סוג הלקוח | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_mac | Mac – הMAC של המחשב | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_rename | Status – הצליח/לא הצליח (Boolean)  Location – מיקום הקובץ  New\_name – השם החדש של הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_delete | Status – הצליח/לא הצליח (Boolean)  Location – מיקום הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_do\_move | New\_path – המיקום החדש של הקובץ  Old\_path – המיקום הישן של הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_status\_move\_to\_client | Status – הצליח/לא הצליח (Boolean)  Moved\_to – המיקום אליו הועבר הקובץ  Moved\_from – המיקום ממנו הועבר הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_move\_to\_server | Status – הצליח/לא הצליח (Boolean)  Old\_path – המיקום הישן של הקובץ  New\_path – המיקום החדש של הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_do\_clone | New\_path – המיקום החדש של הקובץ  Old\_path – המיקום שממנו יש להעתיק את הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_clone\_to\_client | Status – הצליח/לא הצליח (Boolean)  Cloned\_to – המיקום אליו שוכפל הקובץ  Cloned\_from – המיקום ממנו שכפלו את הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_status\_clone\_to\_server | Status – הצליח/לא הצליח (Boolean)  copy\_to – המיקום אליו שוכפל הקובץ  copy\_from – המיקום ממנו שכפלו את הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_create | Status – הצליח/לא הצליח (Boolean)  Location – מיקום הקובץ  Typ – סוג הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_open\_file | Status – הצליח/לא הצליח (Boolean)  Path – מיקום הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_status\_change\_file | Status – הצליח/לא הצליח (Boolean)  Path – מיקום הקובץ | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| Pack\_file\_tree | Path – המיקום של התיקייה | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |
| pack\_key | Key – מפתח סימטרי | בונה הודעה ע"פ הפרוטוקול ומחזירה packed str |

| fileHandler | | |
| --- | --- | --- |
| מחלקה המקבלת עץ קבצים ומפעילה עליו פעולות | | |
| שם משתנה | תפקיד | |
| My\_ip | הIP של המחשב המריץ את התוכנית | |
| username | שם המשתמש המייצג את האובייקט | |
| User\_path | הנתיב למיקום של הקבצים של המשתמש במחשב | |
| פעולה | טענת כניסה | טענת יציאה |
| Init | Username – שם המשתמש | שומרת משתנים נדרשים לפי שם משתמש ומשיגה את הip של המחשב הנוכחי (Settings.get\_ip()) |
| Create\_root | none | יוצרת את התיקייה בה הפרויקט יעבוד משתמשת  ב\_create hidden\_dir, create |
| Is\_local | Path – נתיב מסויים לקובץ | בודק האם הקובץ מקומי או נמצא במחשב אחר ומחזיר אמת או שקר |
| פעולות סטטיות | | |
| Split\_name\_typ | Name – שם קובץ | מחזיר את הקובץ מחולק ל2 שם הקובץ והtyp שלו |
| Build\_name\_for\_file | File\_tree – המילון של הקבצים  New\_dir\_path – שם התיקייה אליו מועבר הקובץ  Path\_of\_file – הנתיב של הקובץ במיקומו הנוכחי  Adding – ההוספה לשם של הקובץ אם כבר שמו תפוס | מחזיר את השם החדש של הקובץ המתאים לתיקייה |
| Direct\_copy\_file | From\_path – המיקום שממנו מעתיקים את הקובץ  To – המיקום אליו מעתיקים את הקובץ | משכפלת את הקובץ ומחזירה האם השכפול עבד או לא |
| rename | Location – מיקום (str)  new\_name – שם חדש (str) | משנה שם קובץ\תיקייה |
| delete | Location – מיקום (str) | מוחק קובץ\תיקייה |
| move | old\_path – מיקום ישן (str)  new\_path – מיקום חדש (str) | מעביר את הקובץ ממקום ישן למקום חדש |
| Split\_path\_last\_part | Path – הנתיב לפרק | מחזירה בצורה של tuple את הנתיב ללא החלק את האחרון ואת החלק האחרון |
| Insert\_ip | Path – הנתיב  Username – שם המשתמש הנמצא בתוך הנתיב  Ip – הip שעל הפונקציה להכניס לנתיב | מחזירה את הנתיב עם הip בתוכו |
| Extract\_ip | Username – השם משתמש שנמצא בנתיב  Path – הנתיב שבו נמצא הIP | מחזירה את הIP שנמצא בתוך הנתיב |
| Remove\_ip | Username – השם משתמש שנמצא בנתיב  Path – הנתיב שנמצא בו הIP | מחזירה את הנתיב ללא הIP |
| Create | path – מיקום ליצור (str)  typ – סוג יצירה (str) | יוצר קובץ\תיקייה במיקום הנתון |
| open\_file | Location – מיקום קובץ (str) | פותחת קובץ |
| \_create\_hidden\_dir | Path – הנתיב לתיקייה | יוצרת תיקייה מוחבאת |
| Get\_user | Path – הנתיב שנמצא בו שם המשתמש | מחזירה את שם המשתמש |
| change\_content | Location – מיקום קובץ (str)  Data – תוכן קובץ חדש (binary) | משנה את תוכן הקובץ הנתון |
| Get\_path\_tree | Path – המיקום של התיקייה שרוצים ליצור ממנה עץ | מחזירה את תוכן התיקייה בצורה של עץ המיוצג כמחרוזת |

| Process\_handler | | |
| --- | --- | --- |
| מחלקה בעלת פעולות המנתחות תהליכים | | |
| שם משתנה | תפקיד | |
| Image\_types | מחזיק את כל הסיומות האפשריות לתמונות | |
| Default\_for\_type | מילון המחבר בין סיומת לשם התהליך המריץ אותו | |
| פעולה | טענת כניסה | טענת יציאה |
| Get\_all\_pid | Process\_name – השם של התהליך | מחזיר רשימה עם כל התהליכים של הסיומת אשר רצים במחשב |
| Get\_process\_name | File\_path – הנתיב של הקובץ | מחזיר את השם של התהליך המתאים לקובץ |
| Wait\_for\_process\_to\_close | Pid – המספר המייצג את התהליך | מחכה שהתהליך ייסגר |

| Db | | |
| --- | --- | --- |
| המחלקה שאחראית על טיפול במבנה הנתונים | | |
| שם משתנה | תפקיד | |
| Conn | מצביע לבסיס הנתונים | |
| DB\_name | שם בסיס הנתונים | |
| Users\_Tbl | טבלת המשתמשים user\_pass | |
| Mac\_tbl | טבלת הmac user\_mac | |
| Curr | המצביע בתוך בסיס הנתונים על מנת לעשות פעולות | |
| Encryption | מטפל בגיבוב הסיסמאות | |
| פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | none | יוצרת את עצם בסיס הנתונים וקוראת ל \_create\_table |
| \_create\_db | None | יוצר את בסיס הנתונים |
| \_create\_table | none | מתחברת לבסיס הנתונים ויוצרת את טבלת user\_pass אם לא קיימת ואת user\_mac אם לא קיימת |
| \_username\_exists | Username – שם המשתמש | בודק האם שם המשתמש קיים בטבלת user\_pass מחזיר boolean לפי אמת או שקר |
| \_username\_mac\_exists | Username – שם המשתמש  mac – כתובת MAC | בודק האם שם המשתמש קיים בטבלת user\_mac מחזיר boolean לפי אמת או שקר |
| add\_user | Username – שם המשתמש password – סיסמא | מוסיף משתמש לטבלאות אם הוא לא קיים באמצעות \_username\_exists ומחזיר boolean |
| add\_user\_mac | Username – שם המשתמש Mac – כתובת MAC | מוסיף לטבלת user\_mac משתמש וmac ובודק אם קיים באמצעות \_username\_mac\_exists ומחזיר boolean |
| Change\_password | Username – שם המשתמש password – סיסמא | משנה את הסיסמא של המשתמש אם הוא קיים ומחזיר boolean |
| Get\_password | Username – שם המשתמש | מחזיר את הסיסמא של המשתמש אם הוא קיים |
| Get\_macs | Username – שם המשתמש | מחזירה את כל הmacs המיוחסים למשתמש |
| Get\_users | Mac – כתובת mac | מחזירה את כל שמות המשתמשים שמיוחס להם mac כזה |

| asymmetricEncrpytion | | |
| --- | --- | --- |
| המחלקה האחראית על הצפנה א-סימטרית | | |
| שם משתנה | תפקיד | |
| public\_key | מפתח ציבורי | |
| private\_key | מפתח פרטי | |
| פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | Private\_key – מפתח פרטי (מאותחל כnone)  public\_key – מפתח ציבורי (מאותחל כnone) | יוצרת את עצם ההצפנה הא-סימטרית ויוצרת מפתח ציבורי ומפתח פרטי אם לא התקבלו כפרמטרים |
| encrypt\_msg | Msg - הודעה (str)  receiver\_key – המפתח הציבורי של המקבל | מחזירה את ההודעה מוצפנת |
| decrypt\_msg | Msg - הודעה מוצפנת (str) | מחזירה את ההודעה מפוענחת |
| get\_public\_key | none | מחזירה את המפתח הציבורי של העצם |

| symmetricEncrpytion | | |
| --- | --- | --- |
| המחלקה האחראית על הצפנה סימטרית | | |
| שם משתנה | תפקיד | |
| key | המפתח הסימטרי | |
| encryption | עצם של AES המשתמש בkey בשביל הצפנה ופענוח | |
| פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | Key – מפתח סימטרי (מאותחל כnone) | יוצרת את עצם ההצפנה הסימטרית ויוצרת מפתח אם לא התקבל כפרמטר |
| encrypt | data - הודעה (str) | מחזירה את ההודעה מוצפנת |
| decrypt | data - הודעה מוצפנת (str) | מחזירה את ההודעה מפוענחת |
| \_pad | Data – מידע לרפד | מחזירה את המידע מרופד |
| \_\_unpad | Data – מידע להוריד ריפוד | מורידה את הריפוד מהמידע |
| hash | Data – מידע לגבב | פעולה חיצונית - מחזירה את המידע מגובב |

| Server\_Comm | | |
| --- | --- | --- |
| המחלקה האחראית על התקשורת של השרת | | |
| שם משתנה | תפקיד | |
| port | הפורט שהשרת משתמש בו | |
| rcv\_q | תור שבו יועברו ההודעות מserverComm לserverLogic | |
| socket | הsocket בוא יעבור המידע בתקשורת | |
| open\_clients | מילון בו מאוכסנים הsocket של הלקוח ובvalue הip המיוחס לsocket ואובייקט ההצפנה הסימטרית  Open\_clients[socket]=[ip, cryptObj] | |
| a\_encrypt | האובייקט של המפתח הא-סימטרי של השרת | |
| is\_running | האם הmain\_loop של המחלקה רץ | |
| Send\_len | כמה בתים יהיה אורך ההודעות המועברות בתקשורת | |
| Receiving\_files | רשימה של ips האמורים לשלוח קובץ | |
| My\_ip | הIP של המחשב המריץ תהליך זה | |
| פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | Port – הפורט בו עובד השרת  rcv\_q – תור שבו יועברו ההודעות מserverComm לlogic  Send\_len – כמה בתים יהיה אורך ההודעות המועברות בתקשורת | מריצה את \_main\_loop  ומאתחלת משתנים |
| disconnect\_client | Ip – הכתובת של הלקוח אשר צריך לנתק אותו  From\_logic – האם הפעולה נקראה מהלוגיקה | משתמשת ב \_find\_socket\_by\_ip  קוראת ל  \_disconnect\_client  ומודיעה לmainServer באמצעות הrcv\_q  ("close", ip) |
| \_disconnect\_client | client – הסוקט של הלקוח אשר צריך לנתק אותו | מוחקת את הלקוח מכלל הרשימות וסוגרות את הסוקט |
| \_find\_socket\_by\_ip | ip – הכתובת של הלקוח אשר מומר לsocket | מחזירה את הsocket המיוחס לip |
| send | Ip – למי לשלוח את ההודעה  Data – מה לשלוח  Encypt – האם להצפין את ההודעה או לא (Boolean) | שולחת את ההודעה את ההודעה לip בהתאם לפרוקטול (אורך ואז הודעה) |
| Send\_file | Ip – הIP אליו לשלוח את הקובץ  Header – ההקדמה לפני קבלת הקובץ  Data – המידע של הקובץ | שולח את הקובץ ללקוח. משתמש בsend |
| \_key\_exchange \*thread | Client – הסוקט של הלקוח  Ip – הכתובת של הלקוח | שולחת את המפתח הציבורי  מקבלת את המפתח הסימטרי  יוצרת עצם הצפנה  מעדכנת רשימה  Open\_clients[client]=[ip,obj]  קוראת גם ל-\_get\_client\_type() |
| \_Get\_client\_type | Client – הסוקט של הלקוח  Ip – הip של הלקוח  Encryption – עצם ההצפנה של הלקוח | מקבלת מהלקוח את הסוג לקוח ושומרת זאת בopenclients |
| \_main\_loop \*thread | None | מקבלת את כל הלקוחות ומקבלת את כל ההודעות מן הלקוחות ולאחר פענוח ההודעות מעבירה את ההודעות לlogic דרך rcv\_q, קוראת ל\_key\_exchange כאשר לקוח חדש מתחבר |
| close\_server | None | משנה את is\_running לfalse |
| \_Receive\_file \*thread | Client – הסוקט של הלקוח  Header – מה שנדרש לדעת על הקובץ | עושה receive לקובץ, שומרת אותו על המחשב ולאחר מכן שמה את הדברים הנדרשים בrcv\_q לפי פרוטוקול |

| Client\_Comm | | |
| --- | --- | --- |
| המחלקה האחראית על התקשורת של הלקוחות | | |
| שם משתנה | תפקיד | |
| port | הפורט שהשרת משתמש בו | |
| rcv\_q | תור שבו יועברו ההודעות מserverComm לserverLogic | |
| server\_ip | הip של השרת | |
| server | הsocket בוא יעבור המידע בתקשורת | |
| symmetric | אובייקט של ההצפנה סימטרית | |
| Send\_len | כמה בתים יהיה אורך ההודעות המועברות בתקשורת | |
| Client\_type | האם הלקוח הוא כללי או לקוח משתמש | |
| Running | האם הmain\_loop רץ או לא | |
| פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | Server\_ip – הIP של השרת  Port – הפורט בו עובד השרת  rcv\_q – תור שבו יועברו ההודעת clientComm לlogic  Send\_len – כמה בתים יהיה אורך ההודעות המועברות בתקשורת  Client\_type – סוג הלקוח (כללי או משתמש) | מריצה את \_main\_loop ומאתחלת משתנים |
| send | Data – מה לשלוח  key– מפתח ציבורי של השרת (מאותחל כnone) | שולחת את ההודעה את ההודעה לשרת, אם קיבלה מפתח שולחת באמצעות המפתח שקיבלה אחרת שולחת עם ההצפנה הסימטרית |
| Send\_file | Header – ההקדמה לפני קבלת הקובץ  Data – המידע של הקובץ | שולח את הקובץ לשרת. משתמש בsend |
| \_key\_exchange \*thread | none | מקבלת מהשרת את המפתח הציבורי שלו  יוצרת מפתח סימטרי  ושולחת אותו לשרת קוראת ל\_send\_client\_type() ומשתמשת בAsymmetricEncryption |
| \_send\_client\_type | None | שולחת לשרת את סוג הלקוח |
| \_main\_loop \*thread | None | מתחברת לשרת, בהודעה הראשונה קוראת ל\_key\_exchnage ולאחר מכן מקבלת הודעות מהשרת, מפענחת אותן ומעבירה לlogic דרך rcv\_q |
| \_Receive\_file \*thread | Header – מה שנדרש לדעת על הקובץ | עושה receive לקובץ שומרת אותו על המחשב ולאחר מכן שמה את הדברים הנדרשים בrcv\_q לפי פרוטוקול |
| Close | None | סוגרת את הסוקט ומודיעה ללוגיקה |

| mainServer | | |
| --- | --- | --- |
| המחלקה של השרת הראשית האחראית על הלוגיקה | | |
| שם משתנה | תפקיד | |
| rcv\_q | תור שבו יועברו ההודעות מComm לLogic | |
| Server\_comm | אובייקט מסוג serverComm האחראי על התקשורת של השרת | |
| Commands | מילון ששומר במפתחות שלו opcode של פעולות ובvalue את הפונקציה המתאימה לopcode | |
| Ip\_mac | מילון ששומר במפתחות שלו ip של לקוחות המחוברים למערכת ובvalue את הmac שלהם | |
| Db | עצם של בסיס הנתונים | |
| User\_comps | מילון שבו ישמר כמפתח הip של משתמש מחובר ובvalue רשימה של כל הip של המחשבים הדולקים שעבד עליהם בעבר | |
| Username\_ip | מילון שבו שמור שם המשתמש שהתחבר והip הנוכחי שלו | |
| פעולה | טענת כניסה | טענת יציאה |
| Main | None | מוציאה מrcv\_q את ההודעות שקיבלה מלקוחות וקוראת לפעולות מתאימות |
| Handle\_got\_mac | Client\_ip – הip ממנו נשלח הmac  mac – כתובת MAC | שומרת בip\_mac את הip ואת הmac. לאחר מכן קורא לget\_users ומוסיפה לuser\_comps את הip במקומות הנדרשים, משתמשת גם בusername\_ip, בנוסף אם מחובר משתמש אשר היה מחובר לmac שנשלח נבקש מclient\_ip את עץ הקבצים שלו |
| Handle\_register | Ip – הip אליו ישלח התשובה  User – שם משתמש  Password – סיסמה | קוראת לadd\_user מחזירה תשובה בהתאם לip, באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_sign\_in | Ip – הip אליו ישלח התשובה  Username – שם משתמש  Password – סיסמה  Mac – כתובת | קוראת לget\_password ומחזירה תשובה בהתאם ל .ipאם התשובה חיובית קוראת לadd\_user\_mac ולאחר מכן לget\_macs ונבקש מכל הmacs המיוחסים למשתמש את עץ הקבצים שלהם ונוסיף לuser\_comps את הip של המשתמש ובvalue את הip של כל המחשבים הדולקים שעבד עליהם בעבר באמצעות ip\_mac ומה שקיבל מget\_macs. מוסיף גם לusername\_ip את שם המשתמש והIp |
| Handle\_got\_file\_tree | Ip - הip ממנו נשלח עץ הקבצים  File\_tree – עץ הקבצים | שולחת באמצעות עטיפה בפרוטוקול ושליחה עם send למשתמש הרצוי את עץ הקבצים לאחר שהכנסנו את הip במקומות הנדרשים בעץ |
| got\_rename | Ip – הip ממנו נשלח המידע  Location – מיקום הקובץ\תיקייה  New\_name – שם חדש | שולחת לכתובת שמיקומה ניתן בLocation את הlocation לשנות ואת השם החדש באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_rename | Got\_ip – הip שהשרת ביקש ממנו למחוק  Status – האם שינוי השם עבד או לא  Location – המיקום של הקובץ ששונה שמו  New\_name – השם החדש של הקובץ | מודיעה ללקוח שביקש את שינוי השם האם שינוי השם עבד או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| got\_delete | Got\_Ip - הip ממנו ביקשו למחוק  Location – מיקום קובץ\תיקייה | שולחת לכתובת שמיקומה ניתן בLocation את הlocation למחוק את הקובץ\תיקייה באמצעות עטיפה בפרוטוקול ושליחה עם send לאחר מכן |
| Handle\_status\_delete | Got\_ip – הip שהשרת ביקש ממנו למחוק  Status – האם המחיקה עבדה או לא  Location – המיקום של הקובץ שנמחק | מודיעה ללקוח שביקש את המחיקה האם המחיקה עבדה או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| got\_move | Ip – הip ממנו נשלח המידע  Move\_from – מיקום הקובץ  Move\_to– מיקום חדש | שולחת לכתובת שמיקומה ניתן בmove\_from Location את move\_from להעביר לmove\_to באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_move | Got\_ip – הIP שיצר את הקובץ  Status – האם היצירה עבדה  Move\_from – המיקום שממנו העבירו את הקובץ  move\_to – המיקום אליו העבירו את הקובץ | מחזירה ללקוח שביקש את ההעברה האם ההעברה עבדה או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| got\_clone | Ip – הip ממנו נשלח המידע  Copy\_from – המיקום ממנו לשכפל  Copy\_to – המיקום לשכפל אליו | שולחת לכתובת שמיקומה ניתן בcopy\_from את copy\_from לשכפל לcopy\_to באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_clone | Got\_ip – הIP שיצר את הקובץ  Status – האם היצירה עבדה  Copy\_from – המיקום שממנו שכפלו את הקובץ  Copy\_to – המיקום אליו שיכפלו את הקובץ | מחזירה ללקוח שביקש את השכפול האם השכפול עבד או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Got\_create | Ip – הip ממנו נשלח המידע  Location – מיקום יצירה  Typ – סוג היצירה | שולחת לכתובת שמיקומה ניתן בlocation את הlocation ליצור עם הtyp שהתקבל באמצעות עטיפה בפרוטוקול ושליחה עם send לאחר מכן נשלח לip את סטטוס הפעולה |
| Handle\_status\_create | Got\_ip – הIP שיצר את הקובץ  Status – האם היצירה עבדה  Location – המיקום שבו נוצר הקובץ  Typ – סוג הקובץ\תיקייה | מחזירה ללקוח שביקש את היצירה האם היצירה עבדה או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_disconnect | Ip - הip שהתנתק  Called\_by\_server\_comm – האם נקרא על ידי הcomm | שולחת לכל המשתמשים אשר משתמשים בקבצים של IP למחוק אותו מעץ הקבצים ומוחקת את הip מכל המילונים הנדרשים אם נדרש מיידעת את server\_comm |
| Get\_key\_by\_value | Dic – המילון להשיג ממנו את המפתח  Value – הערך שממנו נשיג את המפתח | מחזיר את המפתח לפי הערך במילון |

| mainGeneralClient | | |
| --- | --- | --- |
| המחלקה של הלקוח שאינו בהכרח משתמש האחראית על הלוגיקה | | |
| שם משתנה | תפקיד | |
| Client\_rcv\_q | תור שבו יועברו ההודעות מclient\_Comm לLogic | |
| Server\_rcv\_q | תור שבו יועברו ההודעות מserver\_Comm לLogic | |
| client | אובייקט מסוג client\_comm האחראי על התקשורת עם השרת | |
| File\_server | אובייקט מסוג server\_comm האחראי על התקשורת עם לקוחות המשתמש | |
| Commands | מילון ששומר במפתחות שלו opcode של פעולות ובvalue את הפונקציה המתאימה לopcode | |
| פעולה | טענת כניסה | טענת יציאה |
| Main | None | מאפסת משתנים ומפעילה פעולות ראשוניות כמו rcv\_comm() |
| Rcv\_comm \*thread | Comm – האובייקט של התקשורת  Q - התור שבו נקבל הודעות מהתקשורת | מקבלת מהתור הודעות ומפנה לפעולות הרצויות |
| Send\_mac | None | שולחת את הmac של הלקוח לשרת באמצעות עטיפה בפרוטוקל ושליחה עם send |
| Handle\_status\_mac | Client\_got – הcomm של הלקוח  Status – האם הפעולה הצליחה | אם הפעולה נכשלה סוגרת את הלקוח |
| Handle\_asked\_file\_tree | Client\_got – האובייקט של הcomm  Folder\_name – שם התיקייה שרוצה לקבל השרת | שולחת לשרת את עץ הקבצים באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_rename | Client\_got – האובייקט של הclient\_comm  Location – מיקום הקובץ\תיקייה שרוצים לשנות  New\_name – שם הקובץ\תיקייה החדש | קוראת לrename בfilehandler ומחזירה תשובה לשרת אם הפעולה הצליחה או לא באמצעות עטיפה בפרוטוקל וsend |
| Handle\_delete | Client\_got – האובייקט של הclient\_comm  Location – מיקום הקובץ\תיקייה שרוצים למחוק | קוראת לdelete בfilehandler ומחזירה תשובה לשרת אם הפעולה הצליחה או לא באמצעות עטיפה בפרוטוקל וsend |
| Move\_from\_server | Client\_got – האובייקט של הclient\_comm  Move\_from – מיקום הקובץ שרוצים להעביר  Move\_to – מיקום הקובץ החדש | אם מיקום הקובץ החדש הוא מקומי קוראת לmove בfilehandler ומחזירה תשובה לשרת אחרת מעבירה את תוכן הקובץ באמצעות pear to pear ובפעולה send\_file אל המחשב הרצוי |
| Move\_from\_client | Got\_ip – הIP ממנו נשלח הקובץ  Server – האובייקט של הcomm  Status – האם ההעברה הצליחה  Move\_to – לאיפה הועבר הקובץ  Move\_from – מאיפה הועבר הקובץ | מחזירה ללקוח האם ההעברה של הקובץ עבדה או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_move | Client\_got – האובייקט של הcomm  Status – האם הפעולה עבדה או לא  Moved\_to – המיקום שהועבר אליו הקובץ  Moved\_from – המיקום ממנו הועבר הקובץ | מעבירה לשרת האם העברת הקובץ הצליחה באמצעות עטיפה בפרוטוקול ושליחה עם send |
| clone\_from\_server | Client\_got – האובייקט של הclient\_comm  copy\_from – מיקום הקובץ שרוצים לשכפל  copy\_to – מיקום הקובץ החדש | אם מיקום הקובץ החדש הוא מקומי קוראת לdirect\_copy\_file בfilehandler ומחזירה תשובה לשרת אחרת מעבירה את תוכן הקובץ באמצעות pear to pear ובפעולה send\_file אל המחשב הרצוי |
| Clone\_from\_client | Got\_ip – הIP ממנו נשלח הקובץ  Server – האובייקט של הcomm  Status – האם ההעברה הצליחה  copy\_to – לאיפה הועבר הקובץ  copy\_from – מאיפה הועבר הקובץ | מחזירה ללקוח האם השכפול של הקובץ עבד או לא באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_clone | Client\_got – האובייקט של הcomm  Status – האם הפעולה עבדה או לא  copied\_to – המיקום ששוכפל אליו הקובץ  copied\_from – המיקום ממנו שוכפל הקובץ | מעבירה לשרת האם שפכול הקובץ הצליח באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_create | Client\_got – האובייקט של הcomm  Location – המיקום של היצירה  Typ – סוג הקובץ | יוצר את הקובץ באמצעות Filehandler.create() ומודיע לשרת על הפעולה באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_open\_file | Got\_ip – הip שביקש את תוכן הקובץ  Server – האובייקט של הserver\_comm  Location – מיקום הקובץ | משיגה את תוכן הקובץ ושולחת ללקוח משתמש את תוכן הקובץ באמצעות עטיפה בפרוטוקל ושליחה בsend |
| Handle\_changed\_file | Got\_ip – הIP ששינה את תוכן הקובץ  Server – האובייקט של הserver\_comm  Location – מיקום הקובץ | בודקת אם שמירת הקובץ הייתה תקינה ושולחת ללקוח משתמש אם הפעולה הייתה תקינה באמצעות עטיפה בפרוטוקל ושליחה בsend |
| Handle\_status\_mac | Server – האובייקט של הcomm  Status – הצליח או לא | אם status אז נמשיך בריצת התהליך אחרת נעצור אותו |

| mainUserClient | | |
| --- | --- | --- |
| המחלקה של הלקוח שהוא גם משתמש | | |
| שם משתנה | תפקיד | |
| graphic\_q | תור שבו יועברו ההודעות מהגרפיקה ללוגיקה | |
| File\_tree | מילון המייצג את כל עץ הקבצים | |
| Server\_rcv\_q | תור שבו יועברו הודעות מהלוגיקה אל הComm | |
| Client | הcomm שמתקשר עם השרת | |
| User\_name | שם המשתמש | |
| Handle\_tree | תור שאליו יגיעו כל עצי הקבצים לקראת הצגתם בגרפיקה והוספתם לfile\_tree | |
| Ip\_comm | מחזיק בkey IP של המחשב המתוקשר באמצעות pear to pear ואת האובייקט של התקשורת שמדבר עם המחשב | |
| Downloads | מילון של מיקומים של קבצים המחכים להורדה והמיקום שלהם האמיתי | |
| Frame | האובייקט של הגראפיקה | |
| פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | None | מאפסת משתנים מפעילה את הגראפיקה ואת התהליכונים שמדברים עם הגראפיקה ועם התקשורת |
| Monitor\_file \*thread | File\_path – מיקום הקובץ לנתר  End\_q – תור שבו יועבר משהו כאשר יש לסיים את הניתור | מנתרת את הקובץ וקוראת לsave\_file() כאשר המשתמש שומר את הקובץ |
| Folders\_remove | Path – המיקום של האובייקט | מוחקת את האובייקט מfile\_tree ומודיעה לגרפיקה להוריד אותו |
| Folders\_add | Path – מיקום התייקה של האובייקט  Name – השם של הקובץ  Typ – סוג הקובץ | מוסיפה את האובייקט לfile\_tree ואומרת לגרפיקה להוסיף אותו |
| Folders\_rename | Old\_path – המיקום של האובייקט  New\_name – השם החדש של האובייקט | משנה את שם האובייקט בfile\_tree ומודעה לגרפיקה לשנות את שמו |
| Get\_file\_data | Path – מיקום הקובץ | אם הקובץ מקומי פותח את הקובץ או מעביר אותו למקום הרצוי אם המשתמש הוריד את הקובץ (נמצא בdownloads).  אם הקובץ לא מקומי מבקש באמצעות pear to pear את הקובץ באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_open | Comm – האובייקט של התקשורת  Status – האם העברת המידע של הקובץ עבדה או לא  Path – מיקום הקובץ | פותחת את הקובץ למשתמש אחרת קוראת לcall\_error() |
| Save\_file | From\_path – המיקום של הקובץ עם המידע החדש  To\_path – המיקום של הקובץ שבו שומרים את המידע החדש | אם שמירת הקובץ יכולה להתבצע מקומית עושה זאת אחרת שולחת את המידע של from\_path לip שנמצא בto\_path באמצעות עטיפה בפרוטוקול ושליחה עם send\_file() |
| Handle\_status\_save\_file | Status – אם הפעולה עבדה או לא  Path – מיקום הקובץ | אם היה שגיאה כלשהי בשמירת הקובץ קורא לcall\_error() |
| Visualize\_open\_file | File\_path – המיקום של הקובץ לפתיחה  Comm – התקשורת עם המחשב שעליו נמצא הקובץ המקורי | מתחילה ניתור על הקובץ באמצעות monitor\_file() ובאמצעות המחלקה process\_handler מחכה שהקובץ יסגר ע"י המשתמש וסוגרת את התקשורת עם המחשב שנמצא עליו הקובץ |
| Rcv\_graphic \*thread | Q – התור שממנו מועברות הודעות מהגרפיקה | מקבלת הודעות מהגרפיקה עם משתנים ומפנה לפעולות המתאימות |
| Rcv\_comm \*thread | comm – הcomm שמחובר לשרת | מקבל מהשרת הודעות וקורא לפעולות בהתאם |
| Create | Path – מיקום תיקיית האובייקט  Name – שם האובייקט  Typ – סוג האובייקט | אם הpath מקומי יוצרת את האובייקט אחרת מודיעה לשרת ליצור אותו באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_create | Status – אם הפעולה הצליחה  Location – מיקום היצירה  Typ – סוג היצירה | אם הפעולה עבדה קוראת לfolders\_add אחרת קוראת לcall\_error |
| Move\_file | File\_to\_move – המיקום של הקובץ להעביר  Move\_to – לאן להעביר את הקובץ | אם אפשרי להעביר את הקובץ בצורה מקומית מעבירה אחרת מודיעה לשרת על ההעברה באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_move | Status – האם הפעולה עבדה  move\_from – מאיפה הקובץ הועבר  move\_to – לאן הקובץ הועבר | אם הסטטוס חיובי קוראת לfolders\_add() ולאחר מכן לfolders\_remove() אחרת קוראת לcall\_error() |
| Clone\_file | File\_to\_copy – המיקום של הקובץ לשכפל  copy\_to – מיקום השכפול | אם אפשרי לשכפל את הקובץ בצורה מקומית עושה זאת, אחרת מודיעה לשרת על ההעברה באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_clone | Status – האם הפעולה עבדה  Copy\_from – מאיפה הקובץ הועתק  Copy\_to – לאן הקובץ הועתק | אם הסטטוס חיובי קוראת לfolders\_add() אחרת קוראת לcall\_error() |
| Rename | Path – מיקום הקובץ  New\_name – השם החדש של הקובץ | אם אפשרי לשנות את שם הקובץ בצורה מקומית עושה זאת אחרת מודיעה לשרת על שינוי השם באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_rename | Status – האם הפעולה הצליחה  Location – מיקום הקובץ  New\_name – השם החדש של הקובץ | קוראת לfolders\_rename אם הסטטוס חיובי אחרת קוראת לcall\_error |
| Delete | Path – המיקום של הקובץ | אם אפשרי למחוק את הקובץ בצורה מקומית עושה זאת אחרת מודיעה לשרת על המחיקה באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_delete | Status – אם המחיקה עבדה או לא  Location – מיקום האובייקט שנמחק | אם הפעולה עבדה קוראת לfolders\_remove() אחרת קוראת לcall\_error() |
| Get\_file\_tree \*thread | None | מקבל עץ קבצים מהתור handle\_tree מעדכן את file\_tree ומעביר לגרפיקה את העץ שקיבל לעדכון |
| Do\_register | Password – סיסמה | שולחת לשרת נסיון להתחברות באמצעות עטיפה בפרוטוקול ושליחה עם send |
| Handle\_status\_register | Success – אם הצליח או לא | מראה ללקוח בצורה גראפית אם הצליח או לא |
| Do\_connect | Password - סיסמה | מוציאה את כתובת הMAC ושולחת לשרת את נסיון ההתחברות (אם פעולה זו לא תתבצע נסגור את התהליך של mainUserClinet) |
| Handle\_status\_login | Success – הצליח או לא | מציגה למשתמש בצורה גראפית אם הצליח להתחבר או לא, מאפסת משתנים עם שם המשתמש, מפעילה את get\_file\_tree() |
| Handle\_got\_file\_tree | Data – עץ הקבצים שקיבל מהשרת | מעביר לתור handle\_tree את העץ שקיבל |
| Call\_error | Error – מה לכתוב בהודעת השגיאה | קוראת לsend\_to\_grpahics עם ההודעה. |
| Refresh\_cursor | None | אומרת לגרפיקה לשנות את הסמן לרגיל |
| Send\_to\_graphics | Flag – הדגל של הפעולה בגרפיקה  Params – הפרמטרים להעביר לגרפיקה | קוראת לrefresh\_cursor וקוראת לפעולה בגרפיקה עם הדגל והמשתנים |

| settings | | |
| --- | --- | --- |
| המחלקה המחזיקה פעולות בסיסיות ומשתנים אשר יש לשנות לפני הרצת הפרויקט | | |
| שם משתנה | תפקיד | |
| Server\_ip | הIP של השרת | |
| Server\_port | הפורט של השרת | |
| Root | המיקום בו הקבצים נשמרים מקומית על המחשב | |
| Pear\_port | הפורט בו מתקשרים מחשבים ללא עזרת השרת | |
| Dir\_working\_on | המיקום בו נמצא הפרויקט כרגע | |
| Tmp\_directory\_path | המיקום בו נשמרים קבצי tmp | |
| Pic\_path | המיקום של התמונות בהם אנחנו משתמשים בפרויקט | |
| פעולה | טענת כניסה | טענת יציאה |
| Get\_ip | none | מחזירה את הIP של המחשב שקרא לפעולה |
| Get\_mac\_address | None | מחזירה את הmac של המחשב שקרא לפעולה |

| My\_frame |
| --- |
| המחלקה האחראית על החלפת הפאנלים (הבסיס של הגרפיקה) |

| MainPanel |
| --- |
| המחלקה האחראית על המסך שבו המשתמש בוחר האם להירשם או להתחבר בגרפיקה |

| RegisterPanel |
| --- |
| המחלקה האחראית על ההרשמות של המשתמש בגרפיקה |

| LoginPanel |
| --- |
| המחלקה האחראית על ההתחברות של המשתמש בגרפיקה |

| CreateFileDialog |
| --- |
| המחלקה האחראית על הכנסת קלטים של שמות מהמשתמש בגרפיקה |

| TreeFrame | | |
| --- | --- | --- |
| המחלקה האחראית על הצגת עץ הקבצים בגרפיקה והעברת נתונים ללוגיקה מהמשתמש | | |
| שם משתנה | תפקיד | |
| Tree | האובייקט של העץ | |
| File\_commands | כל הפקודות שאפשר לעשות על קבצים | |
| Folder\_commands | כל הפקודות שאפשר לעשות על תיקיות | |
| Root | האובייקט שמייצג את ראש העץ | |
| Command\_q | התור שמתקשר עם הלוגיקה | |
| Folders | רשימה של המיקומים של התיקיות בעץ | |
| Image\_size | המשתנה האחראי על גודל האייקונים | |
| Item\_image\_path | מילון שהמפתח שלו הוא איבר בעץ והערך שלו הוא המיקום של התמונה המתאימה לו | |
| Image\_list | רשימה שבה שמורים כל התמונות של האייקונים הנוכחיים | |
| Path\_item | מילון שהמפתח שלו הוא מיקום של אובייקט והאיבר המתאים לו | |
| Forbidden | אוסף של כל התווים האסורים בשמות של אובייקטים | |
| פעולה | טענת כניסה | טענת יציאה |
| Refresh\_cursor | none | משנה את העכבר למצבו הרגיל |
| Send\_to\_logic | Command – הפקודה ללוגיקה  Data – איזה משתנים מועברים עם הפקודה | מעבירה את העכבר לטעינה ומעבירה בcommand\_q ללוגיקה |
| Change\_size | Evt – האירוע שהפעיל את הפעולה | משנה את גודל העץ |
| Add\_pic | Item – אובייקט בעץ  Name – השם הבסיסי של האובייקט  Is\_folder – האם האובייקט הוא תיקייה | משיגה לאובייקט שם של תמונה להצמיד וקוראת לput\_pic\_for\_item() |
| Put\_pic\_for\_item | Pic\_path – המיקום של התמונה  Item – האובייקט להצמיד לו את התמונה | מצמידה לאובייקט את התמונה ומוסיפה לitem\_image\_path |
| On\_expanded | Evt – האירוע שהפעיל את הפעולה | מראה למשתמש את מה שיש בתיקייה ומשנה את תמונת התיקייה באמצעות put\_pic\_for\_item |
| On\_collapsed | Evt – האירוע שהפעיל את הפעולה | מחביא את תוכן התיקייה ומשנה את תמונת התיקייה באמצעות put\_pic\_for\_item |
| Copy\_paste | Evt – האירוע שהפעיל את הפעולה | בודקת האם המשתמש העתיק או הדביק (באמצעות ctrl + C או ctrl + V) אם הדביק מודיעה ללוגיקה אם העתיק שומרת את ההעתקה |
| On\_drag | Evt – האירוע שהפעיל את הפעולה | משנה את העכבר לתמונה של הקובץ ושומרת את את האובייקט שהמשתמש מעביר |
| On\_drop | Evt – האירוע שהפעיל את הפעולה | מודיעה ללוגיקה מאיפה לאיפה המשתמש העביר קובץ |
| Valid\_input | File\_name – שם הקובץ  Is\_folder – אם האובייקט תיקייה | מחזירה אם המשתמש הכניס קלט תקין או לא לשם של האובייקט |
| Create\_file\_dialog | Command – איזה פעולה מחכה לקלט מהמשתמש  Path – המיקום של האובייקט עליו לחץ המשתמש | בונה מקום בו המשתמש יכול להכניס את הקלט באמצעות CreateFileDialog ושולחת ללוגיקה את מה שהמשתמש הכניס |
| Rename\_object | Old\_path – המיקום הקודם של האובייקט  New\_name – השם החדש של האובייקט | משנה את שם האובייקט בכל הרשימות והמילונים ובעץ |
| Delete\_object | Path – מיקום הקובץ | מוחקת את האובייקט ומוציאה אותו מpath\_item |
| Add\_object | Path – מיקום האובייקט  Name – שם האובייקט  Typ – סוג האובייקט | מוסיפה לרשימות והמילונים המתאימים את האובייקט ומוסיפה לעץ. קוראת לadd\_pic |
| On\_right\_click | Evt – האירוע שהפעיל את הפעולה | מראה למשתמש את אפשרויות הבחירה ולאחר בחירתו קוראת לcommand\_selected() |
| Command\_selected | Evt – האירוע שהפעיל את הפעולה  Item – האיבר שנחברה עליו הפקודה | מעבירה ללוגיקה באמצעות send\_to\_logic את הערכים המתאימים לפי הפקודה שנחברה ע"י המשתמש |
| Convert\_to\_tree | Dic – המילון המייצג עץ  Father – האבא הנוכחי של האיברים בעץ | בצורה רקורסיבית עובר על המילון ומכניס לעץ את העץ שמייצג המילון. קורא לadd\_pic() |
| On\_double\_clicked | Evt – האירוע שהפעיל את הפעולה | אם האובייקט שנלחץ עליו הוא תיקייה מחביא או מראה את תוכן התיקייה, אם האובייקט הוא קובץ פותח אותו |

1. תיאור סביבת הפיתוח –

* שפת התכנות שנבחרה
* לכתיבת הפרויקט – פייתון,
* Wxpython - למימוש הגרפיקה
* - sql בשביל עבודה מול בסיס נתונים
* כלי פיתוח וכלי בדיקה – pycharm: פיתוח בסביבת עבודה שתומכת בפייתון, dbbrowser: בדיקת בסיס הנתונים; לראות שהשינויים שקורים בבסיס הנתונים רצויים, wireshark: בדיקה של התקשורת ברשת לדוגמה לבדוק שהתקשורת מוצפנת, fileexplorer: לבדוק שהקבצים נשמרים כראוי

1. תיאור האלגוריתמים המרכזיים בפרויקט:

**הבעיה: איך לבנות את רשימת הקבצים הקיימים במערכת כדי להציגה למשתמש שהתחבר.**

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | עם התחברות משתמש למערכת השרת פונה לכלל המחשבים ומבקש את רשימת הקבצים שנמצאים אצלו | יעילות – רק בעת הדרישה בונים את הרשימה | זמן תגובה יכול להתארך - מרגע שהמשתמש התחבר עד שהוא רואה את הרשימה |
| 2 | כל מחשב שמתחבר למערכת (בלי קשר להימצאות משתמש) מעביר לשרת את רשימת הקבצים שנמצאים אצלו | זמן תגובה קצר – בשרת תמיד רשימת קבצים עדכנית מה שמקצר את זמן התגובה למשתמש | יעילות – כל הזמן מתבצע תהליך ברקע של העברת רשימה גם אם אין משתמש  עומס על הרשת – העברת רשימת קבצים גם ללא בקשה אפילו אם אין משתמש מעמיסה את הרשת.  בנוסף השרת אוסף את המידע של כלל המשתמשים ללא צורך ומעמיס על התקשורת |

חלופה 1 היא החלופה הרצויה – אומנם היא תיקח בטעינת המערכת אך לא תפגע בביצועי השרת.

**הבעיה: איך לשלוח למשתמש את מבנה רשימת הקבצים בתקשורת?**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | חלופה | יתרונות | חסרונות |
| 1 | כאשר המשתמש מתחבר השרת ישלח לו כל פעם בחלקים את הקבצים, כל פעם שהמשתמש ייכנס לתיקייה השרת ישלח לו את תוכן התיקייה | מונע סרבול – הפרוטוקול תקשורת הופך להרבה יותר קצר ופשוט | זמן תגובה ארוך – כל פעם שילחץ על תיקייה ייקח זמן עד שיופיע לו את תוכן התיקייה.  עומס על הרשת – נוצר יותר תקשורת ברשת |
| 2 | כאשר המשתמש מתחבר השרת ישלח לו את כל מערכת הקבצים בשליחה אחת | זמן תגובה קצר – השרת ישלח למשתמש בשליחה אחת ולא יצטרך לשלוח כל פעם שהמשתמש מבקש תוכן תיקייה. | פרוטוקול מסובך – כדי שנוכל לשלוח מערכת שלמה של קבצים במחרוזת אחת נצטרך לבנות פרוטוקול מסובך. |

חלופה 2 היא החלופה הרצויה – בפרויקט של יש עדיפות לנוחות המשתמש בדגש על רשימת הקבצים שהיא מהות המערכת.

**הבעיה: איזו הצפנה להשתמש?**

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | הצפנה סימטרית  הצפנה בה משתמשים באותו המפתח להצפנה ולפענוח | פשוט מהיר ויעיל | פרוצה לחלוטין, לכל אדם יש את המפתח |
| 2 | הצפנה א-סימרית  לפרט | מאובטח | ההצפנה עצמה לוקחת הרבה זמן מכיוון שהמפתחות יותר ארוכים |
| 3 | הצפנה היברידית לפרט | מאובטח ויחסית מהיר | הצפנת המפתחות ארוכה |

חלופה 3 היא החלופה הרצויה – נרצה בפרויקט זה תקשורת בטוחה אך מהירה.

**הבעיה: איך מעבירים את הקבצים ברשת? מהמחשבים בהם המשתמש עבד בעבר למחשב הנוכחי שלו**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | חלופה | יתרונות | חסרונות |
| 1 | העברה דרך שרת | נוחיות – נשתמש בסוקט אחד במהלך כל התקשורת ברשת | איטיות ועומס על השרת – בסוקט יועבר הרבה מאוד מידע לכן העברת דברים ברשת תהיה תהליך איטי |
| 2 | תקשורת pear to pear באמצעות פורט ספציפי | מהירות – כל פעם שנעביר את המידע הוא לא יעבור דרך השרת אלא ישר בpear to pear | יכול להיווצר עומס על לקוח כאשר מבוקשים ממנו קבצים גדולים |

חלופה 2 היא החלופה הרצויה – בפרויקט זה אנו מתייחסים לעבודה במעבדה של תלמידים לכן לא יבוקשו קבצים גדולים (כמו סרטים) אלא קבצים קטנים (כמו קבצי py, java) לכן חלופה מספר 2 הרבה יותר מהירה.

**הבעיה: איך מציגים למשתמש את מראה הקבצים?**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | חלופה | יתרונות | חסרונות |
| 1 | Windows explorer  (כדי שהקבצים לא יפתחו בצורה ישירה ניצור קבצי דמה, קבצי bat שיריצו את התוכנית הרצויה שינסו לפתוח אותם) | אין צורך בהכנת GUI חדש, אלא השתמשות בwindows explorer שכבר מוכן. | קושי לעבוד עם המערכת של הקבצים בדרך הרצויה.  נראות גראפית לא נוחה למשתמש מכיוון שהicons של הקצים יהיו לא של הקבצים האמיתיים |
| 2 | GUI שאני אצור | קל מאוד לעבוד עם המערכת של הקבצים בדרך הרצויה | יש צורך להכין GUI חדש |

חלופה 2 היא החלופה הרצויה מכיוון שלעבוד עם הwindows explorer יהיה קשה מאוד ובנוסף נראות המערכת היא חשובה מאוד בפרויקט שלנו מכיוון שהוא מיועד לנוחות המשתמש.

1. תיאור מסכי הפרויקט:

**מסך ראשוני** – בחירה אם התחברות או הרשמה.

**מסכי הרשמה\התחברות** - הכנסת שם משתמש וסיסמה אם השם משתמש או סיסמה לא תקינים תוצג הודעה מותאמת.

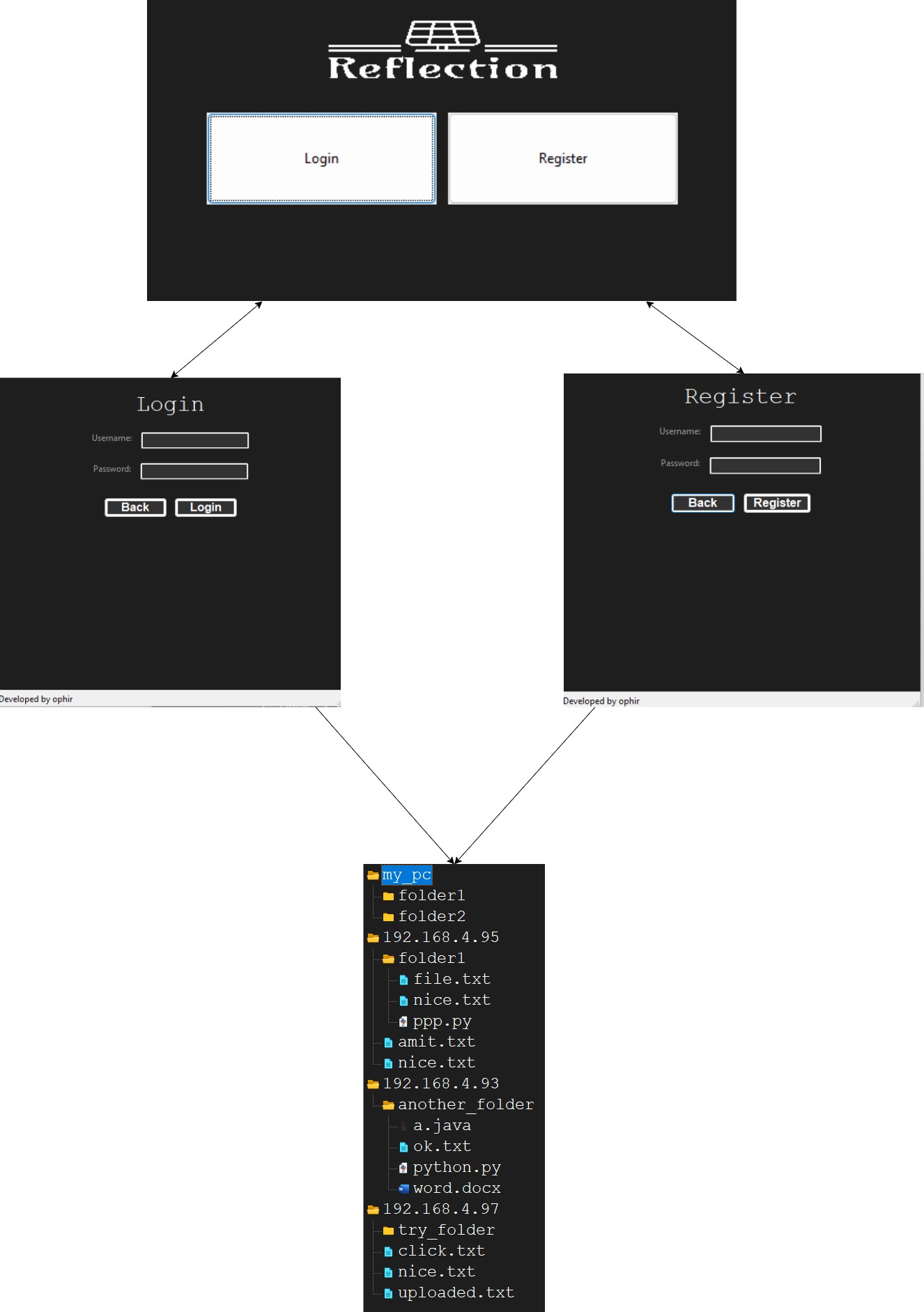
**מסך בית** – לאחר הרשמה או התחברות יפתח מסך בית

**מראה קבצים** - יהיה באפשרות מסך הבית להיכנס למראה הקבצים

תכנון:



בפועל:



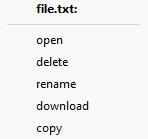
מסכים נוספים:

**מסך בחירת פעולה בתיקייה:**

**A screenshot of a computer

Description automatically generated**

**מסך בחירת פעולה בקובץ:**

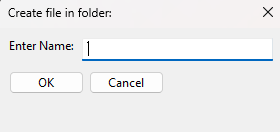
****

**מסך שינוי שם:**

**A screenshot of a computer

Description automatically generated**

**מסך יצירת קובץ:**

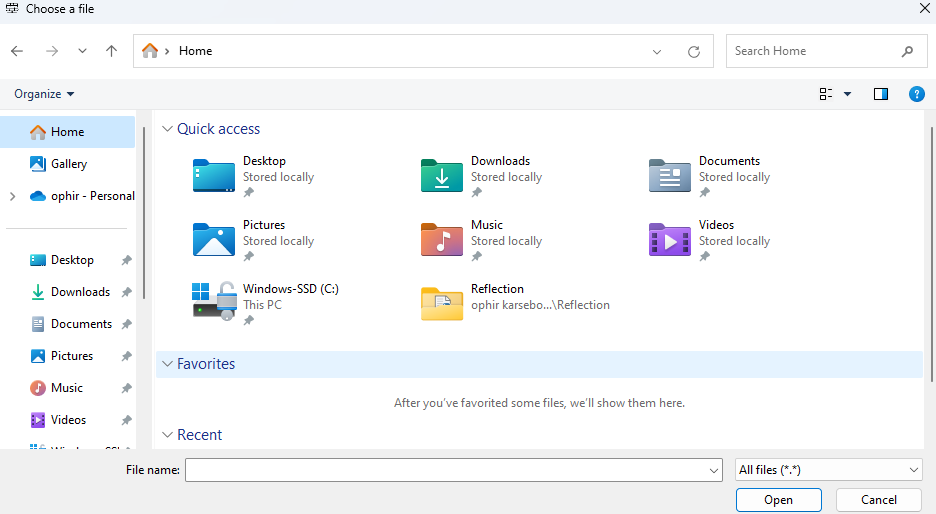
****

**מסך שגיאה כללי:**

**A white rectangular object with black text

Description automatically generated**

**מסך בחירת העלאת קובץ\מיקום להורדת קובץ:**

****

1. תיאור פרוטוקול התקשורת

**כל ההודעות בתקשורת מוצפנות באמצעות מפתח סימטרי שמועבר בתחילת התקשורת (אפשר לראות בפרוטוקול החלפת מפתחות)**

פרוטוקול לקוח משתמש לשרת

אורך כל הודעה ישלח לפני כל ההודעה, אורכו 6 בתים, לאחר מכן ישלח Opcode שאורכו 2 בתים ראשונים לאחר מכן כל משתנה יופרד ע"י פסיק

| הודעות | מ- | אל | שדות-גודל | opcode | דוגמה |
| --- | --- | --- | --- | --- | --- |
| הרשמה | לקוח | שרת | שם משתמש,סיסמה | 01 | 01ophir,12345 |
| סטטוס הרשמה | שרת | לקוח | סטטוס | 02 | 02ok  02no |
| התחברות | לקוח | שרת | שם משתמש\_סיסמה  mac | 03 | 03ophir,12345, AC:BD:9D:F8:35:B2 |
| סטטוס התחברות | שרת | לקוח | סטטוס | 04 | 04ok  04no |
| רשימת קבצים | שרת | לקוח | יפורט בהמשך | 05 | יפורט בהמשך |
| יצירת קובץ\תיקייה | לקוח | שרת | מיקום, סוג | 06 | 062.2.2.2/new\_file,txt  062.2.2.2/new\_folder,fld |
| סטטוס יצירה | שרת | לקוח | מצב,מיקום,סוג | 07 | 07ok,1.1.1.1/file,txt |
| שינוי שם | לקוח | שרת | מיקום, שם חדש | 08 | 081.1.1.1/folder/file.txt,text.txt |
| סטטוס שינוי שם | שרת | לקוח | מצב,מיקום,שם | 09 | 09ok,1.1.1.1/file,new\_file |
| מחיקה | לקוח | שרת | מיקום | 10 | 102.2.2.2/file.txt |
| סטטוס מחיקה | שרת | לקוח | מצב,מיקום | 11 | 11ok,1.1.1.1/file |
| העברה | לקוח | שרת | מיקום,מיקום | 12 | 124.4.4.4/file.txt,4.4.4.4/folder |
| סטטוס העברה | שרת | לקוח | מצב,מיקום,מיקום | 13 | 13ok,1.1.1.1/fold/file,1.1.1.1/fold1 |
| שכפול | לקוח | שרת | מיקום,מיקום | 14 | 142.2.2.2/file.txt,4.4.4.4/folder |
| סטטוס שכפול | שרת | לקוח | מצב,מיקום,מיקום | 15 | 15ok,1.1.1.1/fold/file,1.1.1.1/fold |
| הורדת תיקיית מחשב | שרת | לקוח | שם תיקייה | 30 | 302.2.2.2 |
| אם משתמש או כללי | לקוח | שרת | כללי או משתמש - 1 | 35 | 35U |

פרוטוקול לקוח משתמש ללקוח בעל קובץ

אורך כל הודעה ישלח לפני כל ההודעה, אורכו 8 בתים, לאחר מכן ישלח Opcode שאורכו 2 בתים ראשונים לאחר מכן כל משתנה יופרד ע"י פסיק

| הודעות | מ- | אל | שדות-גודל | opcode | דוגמה |
| --- | --- | --- | --- | --- | --- |
| פתיחת קובץ | משתמש | בעל קובץ | path | 16 | 161.1.1.1/file.txt |
| תשובה לפתיחת קובץ | בעל קובץ | משתמש | אישור-2, מיקום קובץ, אורך מידע, מידע  (אם יש) | 17 | 17ok,1.1.1.1/file.txt ,len,data  17no |
| שינוי תוכן קובץ | משתמש | בעל קובץ | מיקום קובץ,אורך קובץ, מידע | 18 | 184.4.4.4/file.txt,len,data |
| סטטוס שינוי תוכן קובץ | בעל קובץ | משתמש | סטטוס | 19 | 19ok |

פרוטוקול לקוח שאינו בהכרח משתמש לשרת

אורך כל הודעה ישלח לפני כל ההודעה, אורכו 6 בתים, לאחר מכן ישלח Opcode שאורכו 2 בתים ראשונים לאחר מכן כל משתנה יופרד ע"י פסיק

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| הודעות | מ- | אל | שדות-גודל | opcode | דוגמה |
| בקשת רשימת קבצים | שרת | לקוח | שם התיקייה שהשרת מבקש | 31 | 31ophir |
| רשימת קבצים | לקוח | שרת | יפורט בהמשך | 20 | יפורט בהמשך |
| שינוי שם | שרת | לקוח | מיקום, שם חדש | 21 | 211.1.1.1/folder/file.txt,text.txt |
| סטטוס שינוי שם | לקוח | שרת | סטטוס,מיקום,שם חדש | 22 | 22ok,1.1.1.1/file,1.1.1.1/file2 |
| מחיקה | שרת | לקוח | מיקום | 23 | 232.2.2.2/file.txt |
| סטטוס מחיקה | לקוח | שרת | סטטוס,מיקום | 24 | 24ok,1.1.1.1/file |
| העברה | שרת | לקוח | מיקום,מיקום | 25 | 254.4.4.4/file.txt,4.4.4.4/folder |
| סטטוס העברה | לקוח | שרת | סטטוס,קובץ,מיקום | 26 | 26ok,1.1.1.1/file,1.1.1.1/folder |
| שכפול | שרת | לקוח | מיקום,מיקום | 27 | 272.2.2.2/file.txt,4.4.4.4/folder |
| סטטוס שכפול | לקוח | שרת | סטטוס,מיקום,מיקום | 28 | 28no,1.1.1.1/file,1.1.1.1/folder |
| שליחת mac | לקוח | שרת | Mac | 29 | 29AC:BD:9D:F8:35:B2 |
| סטטוס שליחת mac | שרת | לקוח | סטטוס | 34 | 34no |
| יצירה | שרת | לקוח | מיקום,סוג | 32 | 321.1.1.1/folder,fld |
| סטטוס יצירה | לקוח | שרת | סטטוס, מיקום, סוג | 33 | 33no,1.1.1.1/folder,fld |
| אם משתמש או כללי | לקוח | שרת | כללי או משתמש - 1 | 35 | 35G |

פרוטוקול לקוח כללי ללקוח כללי

אורך כל הודעה ישלח לפני כל ההודעה, אורכו 8 בתים, לאחר מכן ישלח Opcode שאורכו 2 בתים ראשונים לאחר מכן כל משתנה יופרד ע"י פסיק

| הודעות | מ- | אל | שדות-גודל | opcode | דוגמה |
| --- | --- | --- | --- | --- | --- |
| העתקת קובץ | שולח קובץ | שומר קובץ | New\_path,  Old\_path | 36 | 361.1.1.1/file.txt,  2.2.2.2/file.txt |
| תשובה להעתקת קובץ | שומר קובץ | שולח קובץ | Status,new\_path,old\_path | 37 | 37ok1.1.1.1/file.txt,  2.2.2.2/file.txt |
| העברת קובץ | שולח קובץ | שומר קובץ | New\_path,  Old\_path | 38 | 381.1.1.1/file.txt,  2.2.2.2/file.txt |
| תשובה להעברת קובץ | שומר קובץ | שולח קובץ | Status,new\_path,old\_path | 39 | 39ok1.1.1.1/file.txt,  2.2.2.2/file.txt |

פרוטוקול החלפת מפתחות

אורך כל הודעה ישלח לפני כל ההודעה, אורכו 3 בתים, לאחר מכן ישלח Opcode שאורכו 2 בתים ראשונים לאחר מכן כל משתנה יופרד ע"י פסיק

| הודעות | מ- | אל | שדות-גודל | opcode | דוגמה |
| --- | --- | --- | --- | --- | --- |
| מפתח א-סימטרי  (לא מוצפן) | שרת | לקוח | מפתח א-סימטרי פומבי | 98 | 98key |
| מפתח סימטרי  (מוצפן באמצעות המפתח הציבורי) | לקוח | שרת | מפתח סימטרי | 99 | 99key |

הסבר על העברת מערכת קבצים:

**Opcode: 05**

הסבר: יישלח מחרוזת המתארת את מערכת הקבצים בצורה הזו:

Full path של התיקייה לאחר מכן סימן שאלה לאחר מכן בסוגריים מרובעות בצורה של רשימה כל התיקיות שנמצאות בתוך הfull path לאחר מכן סימן שאלה ולבסוף בסוגריים מרובעות בצורה של רשימה כל הקבצים שנמצאים בתוך full path. כל זה יבוצע בצורה רקורסיבית עבור כל התיקיות במערכת הקבצים

מ – שרת

אל – לקוח

שדות: full\_path?[folders]?[files]

דוגמה:

protocol\_example?['dir1', 'dir2', 'dir3']?[] protocol\_example\dir1?['folder\_in\_1']?['another\_text.txt', 'text.txt']

protocol\_example\dir1\folder\_in\_1?[]?[]

protocol\_example\dir2?['hello\_in\_2']?[]

protocol\_example\dir2\hello\_in\_2?[]?['hello.txt', 'ok.txt']

protocol\_example\dir3?[]?['archive.rar', 'image.bmp'],,1.1.1.1

**Opcode: 20**

הסבר: יישלח מחרוזת המתארת את מערכת הקבצים שנמצאת אצל הלקוח בצורה הזו:

Full path של התיקייה לאחר מכן סימן שאלה לאחר מכן בסוגריים מרובעות בצורה של רשימה כל התיקיות שנמצאות בתוך הfull path לאחר מכן סימן שאלה ולבסוף בסוגריים מרובעות בצורה של רשימה כל הקבצים שנמצאים בתוך full path. כל זה יבוצע בצורה רקורסיבית עבור כל התיקיות במערכת הקבצים

מ – לקוח

אל – שרת

שדות: full\_path?[folders]?[files]

דוגמה:

protocol\_example?['dir1', 'dir2', 'dir3']?[] protocol\_example\dir1?['folder\_in\_1']?['another\_text.txt', 'text.txt']

protocol\_example\dir1\folder\_in\_1?[]?[]

protocol\_example\dir2?['hello\_in\_2']?[]

protocol\_example\dir2\hello\_in\_2?[]?['hello.txt', 'ok.txt']

protocol\_example\dir3?[]?['archive.rar', 'image.bmp'],,1.1.1.1

1. תיאור מבני הנתונים

נשתמש בפרויקט בבסיס נתונים שיקרא reflection\_db

נמצא בבסיס נתונים זה 2 טבלאות:

* User\_pass – שומרת סיסמאות לפי משתמש סיסמאות נשמרות כhash (user = primary key)
* User\_mac – שומרת mac של מחשבים שהמשתמש עבד עליהם לפי משתמש (user+mac = primary key)

User\_pass

|  |  |
| --- | --- |
| Password – varchar(30) | User – varchar(30)(PK) |
| 1234 | Imriking123 |
| 555555 | ophir |
| Abc\_123 | yotam |

User\_mac

|  |  |
| --- | --- |
| Mac – varchar(17)(PK) | User – varchar(30)(PK) |
| 56:ab:56:ac:58:b1 | Imriking123 |
| 57:ab:56:ac:58:b1 | ophir |
| 58:ab:56:ac:58:b1 | yotam |
| 59:ab:56:ac:58:b1 | ophir |

בנוסף תהיה מחיצה עם השם של הuser בכל מחשב שבה ישמרו כל הקבצים שהמשמש ירצה לשמור לפעמים הבאות שהוא מגיע למעבדה.

1. סקירת חולשות והאיומים

שכבת האפליקציה:

* sql injection – מתקפה זו קורת בפרויקט שבו יש שימוש בסיס הנתונים SQL, כאשר מקבלים קלט מסוים המקושר לבסיס הנתונים לדוגמה שם משתמש וסיסמה יש באפשרות המשתמש להכניס קלט מסוים שישבש את בסיס הנתונים. נטפל בהתקפה זו בכך שנכתוב שאילתות נכונות במחלקה db שלא מאפשרות הכנסה של קוד זדוני, בנוסף גם כאשר נשמור את הסיסמאות בסיס הנתונים נשמור אותם לאחר שגיבבנו אותם ובכך אם לאדם לא רצוי יש גישה לבסיס הנתונים הוא לא יוכל לדעת את סיסמאות המשתמשים.
* תהליך ה login אימות ווידוא – כדי שאנשים שאין להם גישה לקבצים מסוימים לא יקבלו את הגישה, נבצע תהליך login לכל משתמש שכאשר כל משתמש יהיה בעל שם משתמש שונה וסיסמה שרק הוא יודע.
* MITM : התקפת MITM (man in the middle) היא התקפה בה אדם שמאזין בתקשורת רואה מה מועבר בה. נטפל בהתקפה זו בכך שנעבוד עם הצפנה היברידית שתחסום מהאדם הזדוני להבין מה מועבר בתקשורת
* DOS/DDOS : התקפת dos (denial of service) היא התקפה שלקוח משתמש בקצב מאוד מהיר במשאבי השרת ובכך מתיש את השרת ומפריע לו לנהל את כל שאר הלקוחות שלו. נמנע התקפה זו בכך שנראה שיש פעילות לא נורמלית של לקוח אחד נחסום את הip שלו ולא ניתן לו להתחבר לשרת יותר. התקפת ddos (distributed denial of service) היא התקפה בה מבוצע dos מהרבה מחשבים אך התקפה זו מטופלת כאשר טיפלנו בdos מכיוון שהפרויקט שלי נמצא במעבדה קטנה ואין הרבה מחשבים שיכולים להכביד על השרת.

שכבת התעבורה:

* נשתמש בפרוטוקל tcp ובהצפנה היברידית

הפעלת המערכת:

* הפרויקט שלי מושתת על כך שכאשר המחשב נדלק יופעל הקוד של הgeneralClient לכן אדם (בזדון או לא בזדון) יכול למחוק את הקובץ הספציפי הזה מהמחשב. נטפל בבעיה זו בכך שהמחשבים במעבדה יהיו נעולים ומאותחלים. הכוונה היא שכאשר מפעילים את המחשב תמיד הוא יחזור למצבו הקבוע שקבענו, והקבצים שלנו יהיו נעולים ולא יהיה אפשר לשנות אותם.

# פרק ה' - reflection - הקוד

1. עבור (שעושים קטעי קוד מיוחדים משהו מיוחד, משהו מסובך, משהו בדרך שונה, משהו יפה בעיניכם ויש לכלול בתוכם את קטעי הקוד הרלוונטים לאלגוריתמים המרכזיים שציינתם בפרק הקודם):

**הבעיה: איך לבנות את רשימת הקבצים הקיימים במערכת כדי להציגה למשתמש שהתחבר.**

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | עם התחברות משתמש למערכת השרת פונה לכלל המחשבים ומבקש את רשימת הקבצים שנמצאים אצלו | יעילות – רק בעת הדרישה בונים את הרשימה | זמן תגובה יכול להתארך - מרגע שהמשתמש התחבר עד שהוא רואה את הרשימה |
| 2 | כל מחשב שמתחבר למערכת (בלי קשר להימצאות משתמש) מעביר לשרת את רשימת הקבצים שנמצאים אצלו | זמן תגובה קצר – בשרת תמיד רשימת קבצים עדכנית מה שמקצר את זמן התגובה למשתמש | יעילות – כל הזמן מתבצע תהליך ברקע של העברת רשימה גם אם אין משתמש  עומס על הרשת – העברת רשימת קבצים גם ללא בקשה אפילו אם אין משתמש מעמיסה את הרשת.  בנוסף השרת אוסף את המידע של כלל המשתמשים ללא צורך ומעמיס על התקשורת |

חלופה 1 היא החלופה הרצויה – אומנם היא תיקח בטעינת המערכת אך לא תפגע בביצועי השרת.

**בקטע קוד זה השרת שולח לכל המחשבים שהמשתמש התחבר אליהם בקשה שישלחו לו את רשימת הקבצים במחשב שלהם.**

**def** handle\_sign\_in(user\_ip: str, vars: list):  
 *"""  
 signs in the user and returns response accordingly* **:param** *user\_ip: user's ip* **:param** *vars: username, password, mac address* **:return***: None  
 """* print(**'got to sign in'**)  
 **if** len(vars) != 3:  
 handle\_disconnect(user\_ip, **False**)  
 **return** user, password, user\_mac = vars  
 db\_pass = db.get\_password(user)  
 hashed\_pass = symmetrical\_encryption.SymmetricalEncryption.hash(password)  
 **if** db\_pass == hashed\_pass:  
  
 db.add\_user\_mac(user, user\_mac)  
 server\_comm.send(user\_ip, protocol.pack\_status\_login(**True**))  
 macs\_worked\_on = db.get\_macs(user)  
 print(**'macs\_worked\_on:'**, macs\_worked\_on)  
 user\_comps[user\_ip] = [(user\_ip[0], **'G'**)]  
 username\_ip[user] = user\_ip  
  
 *# adding to user comps* **for** mac **in** macs\_worked\_on:  
 **for** ip **in** ip\_mac:  
 **if** ip\_mac[ip] == mac:  
 user\_comps[user\_ip].append(ip)  
 server\_comm.send(ip, protocol.pack\_ask\_file\_Tree(**f'{**user**}'**))  
  
 print(**'user\_comps:'**, user\_comps)  
 *# asking file tree from own user mac* **if** user\_mac **not in** macs\_worked\_on:  
 server\_comm.send((user\_ip[0], **'G'**), protocol.pack\_ask\_file\_Tree(**f'{**user**}'**))  
  
 **else**:  
 server\_comm.send(user\_ip, protocol.pack\_status\_login(**False**))

**הבעיה: איך לשלוח למשתמש את מבנה רשימת הקבצים בתקשורת?**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | חלופה | יתרונות | חסרונות |
| 1 | כאשר המשתמש מתחבר השרת ישלח לו כל פעם בחלקים את הקבצים, כל פעם שהמשתמש ייכנס לתיקייה השרת ישלח לו את תוכן התיקייה | מונע סרבול – הפרוטוקול תקשורת הופך להרבה יותר קצר ופשוט | זמן תגובה ארוך – כל פעם שילחץ על תיקייה ייקח זמן עד שיופיע לו את תוכן התיקייה.  עומס על הרשת – נוצר יותר תקשורת ברשת |
| 2 | כאשר המשתמש מתחבר השרת ישלח לו את כל מערכת הקבצים בשליחה אחת | זמן תגובה קצר – השרת ישלח למשתמש בשליחה אחת ולא יצטרך לשלוח כל פעם שהמשתמש מבקש תוכן תיקייה. | פרוטוקול מסובך – כדי שנוכל לשלוח מערכת שלמה של קבצים במחרוזת אחת נצטרך לבנות פרוטוקול מסובך. |

חלופה 2 היא החלופה הרצויה – בפרויקט של יש עדיפות לנוחות המשתמש בדגש על רשימת הקבצים שהיא מהות המערכת.

**בקטע קוד זה נבנית המחרוזת לפי המיקום של התיקייה.**

**def** get\_path\_tree(path: str):  
 *"""  
 returns path to dir returns tree of dir* **:param** *path: path to dir* **:return***: file tree of path  
 """* file\_tree = **''  
 for** part **in** os.walk(path):  
 part = **f'{**part[0]**}?{**part[1]**}?{**part[2]**}\n'** file\_tree += part  
  
 **return** file\_tree

**בקטע קוד זה השרת מעביר למילון את המחרוזת של עץ הקבצים, לאחר מכן מוסיף ip של הלקוח למקומות הנדרשים, מחזיר למחרוזת שמדמה עץ ושולח ללקוח משתמש.**

**def** handle\_got\_file\_tree(got\_ip, vars: list):  
 *"""* **:param** *got\_ip: the ip got file tree from* **:param** *vars: file tree got from got\_ip* **:return***: None  
 """* file\_tree = vars[0]  
 got\_ok = **True** ip\_to\_send = **None** folders\_with\_ip = {}  
 **if** len(vars) == 1:  
 user\_to\_send = FileHandler.get\_user(list(file\_tree.keys())[0])  
 print(**'user\_to\_send:'**, user\_to\_send)  
 **if** user\_to\_send **in** username\_ip:  
 ip\_to\_send = username\_ip[user\_to\_send]  
 print(**'to\_send:'**, ip\_to\_send)  
  
 **else**:  
 got\_ok = **False** ip\_to\_send = **None  
  
 if** got\_ok **and** ip\_to\_send **in** user\_comps **and** got\_ip **in** user\_comps[ip\_to\_send]:  
  
 folders\_with\_ip = {}  
  
 *# inserting ip to each path in dictionary* **for** folder\_path **in** file\_tree.keys():  
 user\_plus\_ip = os.path.join(user\_to\_send, got\_ip[0])  
 **if** user\_to\_send **in** folder\_path:  
 new\_folder\_path = folder\_path.replace(user\_to\_send, user\_plus\_ip, 1)  
 folders\_with\_ip[new\_folder\_path] = file\_tree[folder\_path]  
 **else**:  
 got\_ok = **False  
 break  
  
 else**:  
 got\_ok = **False  
 else**:  
 got\_ok = **False  
  
 if** got\_ok **and** ip\_to\_send:  
 *# sending file tree to user* server\_comm.send(ip\_to\_send, protocol.pack\_send\_file\_tree(folders\_with\_ip))  
  
 **else**:  
 print(**'error in getting file tree'**)  
 handle\_disconnect(got\_ip, **False**)

**הבעיה: איזו הצפנה להשתמש?**

| מס' | חלופה | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | הצפנה סימטרית  הצפנה בה משתמשים באותו המפתח להצפנה ולפענוח | פשוט מהיר ויעיל | פרוצה לחלוטין, לכל אדם יש את המפתח |
| 2 | הצפנה א-סימרית  לפרט | מאובטח | ההצפנה עצמה לוקחת הרבה זמן מכיוון שהמפתחות יותר ארוכים |
| 3 | הצפנה היברידית לפרט | מאובטח ויחסית מהיר | הצפנת המפתחות ארוכה |

חלופה 3 היא החלופה הרצויה – נרצה בפרויקט זה תקשורת בטוחה אך מהירה.

**החלפת מפתחות צד שרת:**

**def** \_key\_exchange(self, client, ip):  
 *"""  
 exchanging keys with client, puts client in open clients at the end* **:param** *client: client socket* **:param** *ip: ip address* **:return***:  
 """* self.send(client, server\_protocol.pack\_public\_key(self.a\_encrypt.get\_public\_key()), **False**)  
 **try**:  
 length = int(client.recv(self.send\_len).decode())  
 data = client.recv(length)  
 data = self.a\_encrypt.decrypt\_msg(data)  
 **except** Exception **as** e:  
 print(**"error in key\_exchange: "** + str(e))  
 self.disconnect\_client(client)  
 **else**:  
  
 encryption = symmetrical\_encryption.SymmetricalEncryption(data[2:])  
  
 self.\_get\_client\_type(client, ip, encryption)

**החלפת מפתחות צד לקוח:**

**def** \_key\_exchange(self):  
 *"""  
 gets server's public key and sends server symmetric key* **:return***: None  
 """* **try**:  
 length = int(self.server.recv(self.send\_len).decode())  
 data = self.server.recv(length)  
  
 **except** Exception **as** e:  
 self.close()  
 sys.exit(**"server is down, try again later"**)  
  
 public\_key = data[2:]  
 print(**'pulic key:'**, public\_key)  
 self.symmetric = symmetrical\_encryption.SymmetricalEncryption()  
 key\_to\_send = general\_client\_protocol.pack\_key(self.symmetric.key)  
 key\_to\_send = AsymmetricEncryption.encrypt\_msg(key\_to\_send, public\_key)  
 **try**:  
 self.server.send(str(len(key\_to\_send)).zfill(self.send\_len).encode())  
 self.server.send(key\_to\_send)  
 **except** Exception **as** e:  
 self.close()  
 sys.exit(**'could not send server key encrypted'**)  
  
 **else**:  
 self.\_send\_client\_type()

**הבעיה: איך מעבירים את הקבצים ברשת? מהמחשבים בהם המשתמש עבד בעבר למחשב הנוכחי שלו**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | חלופה | יתרונות | חסרונות |
| 1 | העברה דרך שרת | נוחיות – נשתמש בסוקט אחד במהלך כל התקשורת ברשת | איטיות ועומס על השרת – בסוקט יועבר הרבה מאוד מידע לכן העברת דברים ברשת תהיה תהליך איטי |
| 2 | תקשורת pear to pear באמצעות פורט ספציפי | מהירות – כל פעם שנעביר את המידע הוא לא יעבור דרך השרת אלא ישר בpear to pear | יכול להיווצר עומס על לקוח כאשר מבוקשים ממנו קבצים גדולים |

חלופה 2 היא החלופה הרצויה – בפרויקט זה אנו מתייחסים לעבודה במעבדה של תלמידים לכן לא יבוקשו קבצים גדולים (כמו סרטים) אלא קבצים קטנים (כמו קבצי py, java) לכן חלופה מספר 2 הרבה יותר מהירה.

**כדי לגשת לתוכן קובץ שנמצא במחשב אחר יוצר מופע של clientComm ומפעיל תהליכון שמחכה למידע שישלח מהמחשב שהתחבר אליו.**

*# if file is on another computer***else**:  
 ip\_to\_connect = self.file\_handler.extract\_ip(self.user\_name, path)  
 *# setting up pear to pear* **if** ip\_to\_connect **not in** self.ip\_comm:  
 rcv\_q = Queue()  
  
 comm = ClientComm(ip\_to\_connect, Settings.pear\_port, rcv\_q, 8, **'U'**)  
 self.ip\_comm[ip\_to\_connect] = comm  
 threading.Thread(target=self.rcv\_comm, args=(comm,), daemon=**True**).start()  
  
 **else**:  
 comm = self.ip\_comm[ip\_to\_connect]  
  
 comm.send(protocol.pack\_do\_open\_file(path))

**הבעיה: איך מציגים למשתמש את מראה הקבצים?**

|  |  |  |  |
| --- | --- | --- | --- |
| מס' | חלופה | יתרונות | חסרונות |
| 1 | Windows explorer  (כדי שהקבצים לא יפתחו בצורה ישירה ניצור קבצי דמה, קבצי bat שיריצו את התוכנית הרצויה שינסו לפתוח אותם) | אין צורך בהכנת GUI חדש, אלא השתמשות בwindows explorer שכבר מוכן. | קושי לעבוד עם המערכת של הקבצים בדרך הרצויה.  נראות גראפית לא נוחה למשתמש מכיוון שהicons של הקצים יהיו לא של הקבצים האמיתיים |
| 2 | GUI שאני אצור | קל מאוד לעבוד עם המערכת של הקבצים בדרך הרצויה | יש צורך להכין GUI חדש |

חלופה 2 היא החלופה הרצויה מכיוון שלעבוד עם הwindows explorer יהיה קשה מאוד ובנוסף נראות המערכת היא חשובה מאוד בפרויקט שלנו מכיוון שהוא מיועד לנוחות המשתמש.

**קטע קוד זה ממיר מילון לעץ הקבצים בגרפיקה בצורה רקורסיבית.**

**def** convert\_to\_tree(self, dic: dict, father=**None**):  
 *"""  
 gets dictionary and converts it to tree* **:param** *dic: dictionary* **:param** *father: a parent* **:return***: None  
 """* **if not** father:  
 print(**'dictionary working with:'**, dic)  
 father\_path = next(iter(dic.keys()))  
 father = self.root  
 **else**:  
 father\_path = self.tree.GetItemData(father)  
  
 folder = **True  
 if** father\_path **not in** dic:  
 **return** *# runs on every object in this folder adds it to tree and if is folder then* **for** element **in** dic[father\_path]:  
 **if** element == **','**:  
 folder = **False  
 continue** path = os.path.join(father\_path, element)  
 **if** element == self.my\_ip:  
 element = **'my\_pc'** new\_item = self.tree.AppendItem(father, element, data=path)  
  
 self.path\_item[path] = new\_item  
 self.add\_pic(new\_item, element, folder)  
 **if** folder:  
 self.folders.append(path)  
 self.convert\_to\_tree(dic, new\_item)

**קטע קוד זה מפעיל את התהליכון של הניטור פותח את הקובץ וכאשר המשתמש סוגר את הקובץ הפונקציה סוגרת את התהליכון של הניתור.**

def visualize\_open\_file(self, file\_path, comm: ClientComm):  
 *"""  
 opening file to user and activates monitoring on file  
 :param file\_path: path of file  
 :param comm: the client\_comm opened with other pc  
 :return: None  
 """  
  
 # start monitoring file* q = Queue()  
 threading.Thread(target=self.monitor\_file, args=(file\_path, q), daemon=True).start()  
  
 process\_name = process\_handler.get\_process\_name(file\_path)  
 ls1 = process\_handler.get\_all\_pid(process\_name)  
 dir\_path, \_ = FileHandler.split\_path\_last\_part(file\_path)  
 FileHandler.open\_file(file\_path)  
 self.refresh\_cursor()  
 print('ls1:', ls1)  
 *# wait for file to be added to pid list* while True:  
 ls2 = process\_handler.get\_all\_pid(process\_name)  
 print('ls2:', ls2)  
 if ls2 != ls1:  
 break  
  
 new\_pid = set(ls2) - set(ls1)  
 pid = list(new\_pid)[0]  
 print('this is pid:', pid)  
 process\_handler.wait\_for\_process\_to\_close(pid)  
  
 *# closing all related connections to pc* ip = next(filter(lambda item: item[1] == comm, self.ip\_comm.items()), None)[0]  
 del self.ip\_comm[ip]  
 comm.close()  
  
 *# generic action on file to end thread of monitoring file* q.put('end')  
 FileHandler.delete(file\_path)  
  
 print('ended visualize file')

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

1. פירוט הבדיקות

יש לעשות את הבדיקות לאחר התחברות מ2 המחשבים לפחות פעם אחת חוץ מבדיקות 14 ו15.

| **מספר** | **שם הבדיקה (שם שיעיד על תוכן הבדיקה)** | **מה אמורה לבדוק** | **איך מתכננים לבדוק**  **(לתאר בפירוט את שלבי הבדיקה)** | **מתי** | **מה בוצע בפועל** |
| --- | --- | --- | --- | --- | --- |
| 1 | יצירת קבצים | לאחר התחברות למערכת, האם כאשר יוצרים במחשב א' קובץ, הקובץ יימצא במראה הקבצים במחשב ב'. | מתחברים למערכת במחשב א', יוצרים קובץ. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם הקובץ נמצא במראה הקבצים במחשב ב' | 16.5 | נוצר הקובץ במחשב א' והופיע במחשב ב' |
| 2 | מחיקת קבצים | לאחר התחברות למערכת, האם כאשר מוחקים במחשב א' קובץ, הקובץ לא יימצא במראה הקבצים במחשב ב'. | עושים זאת לאחר בדיקה מס' 1.  מתחברים למערכת במחשב א', מוחקים את הקובץ שיצרנו. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם הקובץ נעלם ממראה הקבצים. | 16.5 | נמחק הקובץ במחשב א' והשינוי הופיע במחשב ב' |
| 3 | שינוי שם קובץ | לאחר התחברות למערכת, האם כאשר משנים של קובץ במחשב א', שם הקובץ ישתנה במראה הקבצים במחשב ב'. | עושים זאת לאחר בדיקה מס' 1.  מתחברים למערכת במחשב א', משנים את שם הקובץ שיצרנו. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם שם הקובץ ישתנה במראה הקבצים. | 16.5 | שם הקובץ השתנה במחשב א' והשינוי הופיע במחשב ב' |
| 4 | שינוי תוכן קובץ | לאחר התחברות למערכת, האם כאשר משנים תוכן של קובץ במחשב א', תוכן הקובץ ישתנה במחשב ב'. | עושים זאת לאחר בדיקה מס' 1.  מתחברים למערכת במחשב א', כותבים בתוך הקובץ (txt) כמה מילים. לאחר מכן מתחברים למערכת במחשב ב' ובודקים האם מה שכתבנו בקובץ במחשב א' נמצא בקובץ במחשב ב'. | 16.5 | נפתח הקובץ במחשב א' שונה התוכן שלו. פתחנו את הקובץ במחשב ב' וראינו כי השתנה התוכן |
| 5 | סנכרון פעולות | נבדוק האם כל הפעולות שכתבנו לעיל עובדות כאשר עושים אותם באותו זמן ושום דבר אינו מתבלבל | ניצור 2 קבצים במחשב א' נתחבר למחשב ב' ונבדוק כי שניהם נוצרו. לאחר מכן שוב נתחבר למחשב א' ניצור עוד קובץ, נשנה את שם הקובץ של אחד משני הקבצים ואת תוכן הקובץ ונמחק את הקובץ הנוסף. נתחבר למחשב ב' ונבדוק שכל הפעולות שעשינו התבצעו כראוי | 16.5 | במחשב א' נוצרו 2 הקבצים שם הקובץ הראשון שונה והקובץ השני נמחק. במחשב ב' פתחנו את מראה הקבצים ונראה כי הכל השתנה כראוי |
| 6 | זמני תגובה | האם בעבודה עם קבצים גדולים יש פגיעה בזמני התגובה של המערכת | נשמור 5 קבצים גדולים במחשב ב' דרך מחשב א' ונבדוק האם במחשבים ב' בו מחובר משתמש אחר ברשת יש עיכוב בפעולות רגילות | 16.5 | הקבצים נשמרו במחשב ב' ממחשב א' ולמרות זאת לא היו בעיות בזמני התגובה במחשב ב' |
| 7 | עומס | האם כשיש משתמשים רבים במערכת, הביצועים עדיין טובים | נחבר 4 משתמשים למערכת מ8 מחשבים שונים ונבצע כמה פעולות שמתוארות לעיל ונבדוק האם הפעולות מתבצעות כראוי | 16.5 | חיברנו 4 משתמשים ל8 מחשבים שונים והפעולות התבצעו כראוי אם דיליי קטן מאוד מדי פעם |
| 8 | יציבות | האם המערכת נשארה יציבה אחרי התחברויות והתנתקויות של משתמשים | נחזור על פעולה מס' 7 ומדי פעם נחבר וננתק משתמשים לבדוק האם המערכת אינה קורסת | 16.5 | כאשר מחשבים התנתקו והתחברו המערכת לא קרסה |
| 9 | אבטחה | האם בעת העבודה על הקבצים ממחשב אחר המידע לא זולג החוצה | נפתח wireshark נפתח קובץ של מחשב ב' ממחשב א' ונבדוק האם התוכן שלו מוצג בwireshark | 16.5 | התוכן של הקובץ היה מוצפן כאשר מסתכלים דרך הwireshark |
| 10 | העלאת קובץ | האם כאשר מעלים קובץ אפשרי לפתוח אותו דרך המערכת | מתחברים למחשב מס 1 מעלים קובץ מן המחשב, מתחברים למחשב מס 2 ובודקים אם הקובץ קיים והתוכן שלו נכון | 16.5 | הקובץ עלה והתוכן שלו היה מתאים |
| 11 | הורדת קובץ | האם כאשר מורידים קובץ הוא מופיע במראה הקבצים של ווינדוס | מתחברים למחשב מס 1 מורידים קובץ ובודקים האם הוא נפתח במראה הקבצים של ווינדוס | 16.5 | הקובץ ירד וכאשר הוא נפתח מהwindows file explorer התוכן שלו היה מתאים |
| 12 | העברת קובץ | האם כאשר מעבירים קובץ ממחשב אחד למחשב אחר הקובץ מופיע במחשב האחר | מתחברים למחשב מס 1 מעבירים קובץ מהמחשב למחשב מס 2. בודקים האם הקובץ נמחק מהמחשב. מתחברים למחשב מס 2 ובודקים האם הקובץ נמצא | 16.5 | הקובץ הועבר ממחשב א' למחשב ב' דרך מחשב א'. נראה כי הקובץ נמחק ממחשב א'. התחברנו דרך מחשב ב' וראינו כי הקובץ נמצא. |
| 13 | שכפול קובץ | האם כאשר מעתיקים קובץ ממחשב אחד למחשב אחר הקובץ מופיע במחשב האחר | מתחברים למחשב מס 1 מעתיקים קובץ מהמחשב למחשב מס 2. מתחברים למחשב מס 2 ובודקים אם הקובץ נמצא | 16.5 | הקובץ הועתק ממחשב א' למחשב ב' דרך מחשב א'. התחברנו דרך מחשב ב' וראינו כי הקובץ נמצא |
| 14 | הרשמה למערכת | האם כאשר נרשמים למערכת עם שם משתמש שלא קיים מכניס למערכת הקבצים | נרשמים למערכת עם שם משתמש שלא קיים ובודקים אם נכנס למערכת קבצים. מריצים את המערכת מחדש ומנסים להירשם עם אותו משתמש ובודקים אם מוצגת הודעה ראויה שהמשתמש כבר קיים | 16.5 | נרשמנו למערכת עם שם משתמש שלא קיים ונכנסנו למערכת. יצאנו מהמערכת ניסינו להירשם עם אותו משתמש וקפצה לנו הודעת שגיאה כנדרש |
| 15 | התחברות למערכת | האם כאשר מתחברים למערכת עם שם משתמש קיים מכניס למערכת קבצים | מתחברים למערכת עם שם משתמש קיים ובודקים אם נכנס למערכת הקבצים. מריצים את המערכת מחדש ומנסים להיכנס עם שם משתמש שלא קיים ובודקים שמוצגת הודעה ראויה שהמשתמש לא קיים. | 16.5 | התחברנו למערכת עם שם משתמש קיים והופיע לנו מראה הקבצים. לאחר מכן התנתקנו מהמערכת וניסינו להתחבר עם שם משתמש לא קיים וקפץ הודעת שגיאה כנדרש. |

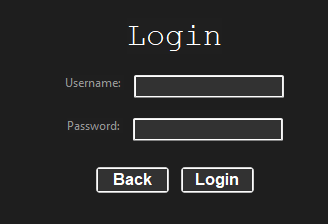
# פרק ז' - reflection – מדריך למשתמש

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

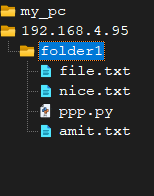
* תחילה יש להיכנס לקובץ ההגדרות settings.py ולשנות את המשתנה server\_ip לip שבו יורץ השרת ובמשתנה root לשנות את המיקום למקום בו רוצים שהקבצים ישמרו בצורה לוקלית ולא ימחקו. (לדוגמה במעבדת בית הספר המקום שנשמרים בו קבצים על המחשב הוא D: .(
* שנית יש להריץ את את קובץ main\_server.py במחשב שהip שלו זהה לserver\_ip. לאחר מכן להריץ בכל מחשב שנפתח את main\_general\_client.py ובמחשבים שבהם נרצה להיות משתמש נריץ את main\_user\_client.py
* על מנת להשתמש במערכת, ראשית יש להירשם. לאחר ההירשמות המשתמש יכול להשתמש במערכת.



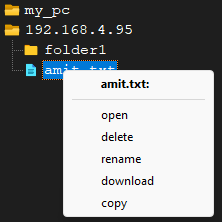
* לאחר הירשמות בפעם הראשונה המשתמש יכול להתחבר.



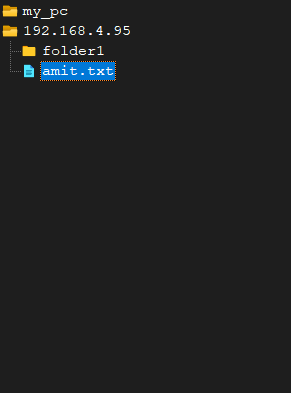
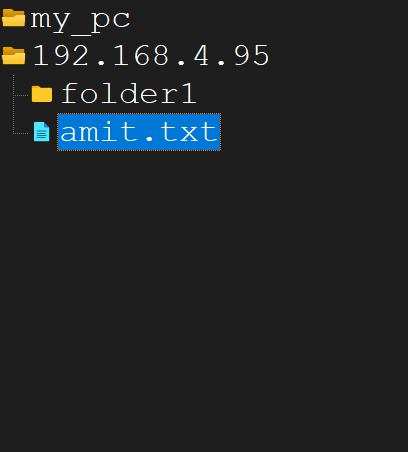
* לאחר הכניסה יוצג עץ הקבצים של המשתמש – כל הקבצים בכל המחשבים: my\_pc המחשב הנוכחי והאחרים מזוהים ע"י ip



* כדי לבצע פעולות בסיסיות על קובץ יש ללחוץ עליו במקש ימני בעכבר ולבחור את הפעולה הרצויה. אותו דבר גם לגבי תיקייה.



כדי להגדיל ולהקטין את המסך יש ללחוץ על CTRL ובאותו זמן לגלגל את הגלגלת.

* העברת קובץ נעשית ע"י גרירת הקובץ ע"י המקש השמאלי בעכבר (במצב לחוץ)

# פרק ח' - reflection – רפלקציה

1. התחלתי את הפרויקט בידיעה שהפרויקט שלי אינו מחדש משהו אלא אני יוצר אותו בשביל שאני אלמד דברים ומתוך סקרנות אישית.

בהחלט הכנת הפרויקט לימדה אותי דברים רבים לדוגמה ניהול תהליכונים מרובים, תקשורת עם הAPI של ווינדוס וגולת הכותרת איך לעבוד עם מערכת קבצים ענקית שמפוזרת בכמה מחשבים.

בפרויקט נתקלתי באתגרים מעניינים ושאלות שלא היו לי אליהם תשובות, לדוגמה איך לסגור תהליכונים שנמצאים בלולאה כאשר אינני יכול לסגור את התהליך שיצר אותם?

נתקלתי בבעיה זאת כאשר הפרויקט שלי נהיה איטי ככל שפתחתי יותר קבצים בפרויקט – כך גיליתי שלא נסגרים אצלי תהליכונים. הבנתי איפה לא נסגרים לי התהליכונים ובכך העברתי אליהם הודעה בתור כאשר לתוכנית אין צורך בהם.

דילמה נוספת שעלתה לי במהלך הפרויקט היא האם לשלוח תמיד בצורה גנרית לשרת פעולות למרות שלפעמים המשתמש מבקש לעשות פעולות מקומיות או לסנן פעולות מקומיות אצל הלקוח. החלטתי בסופו של דבר לסנן פעולות מקומיות אצל הלקוח כדי שהפרויקט יעבוד בצורה היעילה ביותר למרות שלבחור בדרך הגנרית תהיה האפשרות הפשוטה ביותר.

החוויה של כתיבת הפרויקט במעבדה בבית ספר הייתה חוויה מגבשת בין כל חברי הכיתה, לפעמים כשצצה בעיה לבקש עזרה מחברים ולחשוב ביחד עזרו לי להבין איך לי נוח לחשוב על שאלה או בעיה: כאשר ניסיתי להסביר לחברים מה הבעיה שנוצרה הייתי מצייר אותה על הלוח בכיתה בצורה גראפית וככה זה היה עוזר לי להסביר את עצמי בצורה הנכונה ביותר וגם לפעמים להבין את עצמי יותר טוב ובכך לפתור את הבעיה לבד.

בנוסף גם להסתובב בכיתה ולעזור לחברים עם הפרויקט חיבר אותי עם אנשים ובנוסף עזר לי לצאת טיפה מהקופסא עם בעיות שלי כאשר עזרתי לאחרים.

אני חושב שיצירת הפרויקט הכי החוויה הלימודית הכי מעניינת ומסקרנת שהייתה לי, לבנות פרויקט מאפס כאשר המגבלות שלי הן מינימליות לימד אותי הרבה על עצמי ועל איך לנהל את הזמן שלי בצורה הכי נכונה כאשר הפרויקט פרוס על פרק זמן ארוך.

הייתי רוצה להגיד תודה לחברים לכיתה שתמיד היו שמחים לעזור כאשר ביקשתי, לאח שלי שעזר לי לחשוב על דרכים לפתירת בעיות, וכמובן תודה ענקית למרי המורה שלי שתמכה, הציבה גבולות שהייתה צריכה ובמיוחד אני מודה על ההשקעה הענקית בכל תלמיד, שמעתי לפחות סיפור אחד מכל חבר לכיתה על איך מרי פתאום כתבה בהודעה, אחרה"צ, לא בשעות בית ספר, על איזה דרך שהיא מצאה לפתרון לבעיה שדיברו עליה בבית ספר.

אני מאוד מרוצה מהפרויקט שיצא לי, אם היה לי זמן הייתי מוסיף עוד פיצ'רים מיוחדים כגון: העברת תיקיות, שימוש יותר נוח דרך המערכת קבצים של ווינדוס ושיפור הגרפיקה הכללית, למרות זאת אני חושב שבתווך זמן שהיה לי הוצאתי את המקסימום שלי לפרויקט ואני מרוצה מאוד.

# פרק ט' - reflection – ביבליוגרפיה

עבודה עם גרפיקה:

<https://chatgpt.com/>

שאלות כלליות בכל הנושאים:

[stack overflow](https://try.stackoverflow.co/explore-teams?utm_source=adwords&utm_medium=ppc&utm_campaign=kb_teams_search_brand_emea-dach&_bt=657236278309&_bk=stack+overflow&_bm=p&_bn=g&gad_source=1&gclid=CjwKCAjwo6GyBhBwEiwAzQTmc-LfBvmk4dlyh1etNB08bUBw5FSn8mofQYDMQ9zDXsw5xEMk_iqA4BoClhoQAvD_BwE)

שאלות בסיסיות בpython וsql:

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

# קוד הפרויקט

**Settings.py:**

from uuid import getnode  
import socket  
import os  
  
class Settings:  
 server\_ip = '10.100.102.100'  
 server\_port = 2000  
 root = 'C:'  
 root += '\\reflection\\'  
 pear\_port = 2500  
 dir\_working\_on = os.path.dirname(os.path.realpath(\_\_file\_\_))  
 tmp\_directory\_name = 'localChange\\'  
 tmp\_directory\_path = f'{root}{tmp\_directory\_name}'  
 pic\_path = os.path.join(dir\_working\_on, 'graphics\\icons\\')  
 icon\_path = pic\_path + 'icon\_black.png'  
 @staticmethod  
 def get\_ip():  
 *"""  
 returns ip  
 """* s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
 s.connect(("8.8.8.8", 80))  
 ip = s.getsockname()[0]  
 s.close()  
 return ip  
  
 @staticmethod  
 def get\_mac\_address():  
 *""" returns mac address"""* return ':'.join(['{:02x}'.format((getnode() >> i) & 0xff) for i in range(0, 8 \* 6, 8)][::-1])

**General\_client\_protocol.py:**

from Reflection.local\_handler.file\_handler import FileHandler  
  
def unpack(data: str):  
 *"""  
 parsing by protocol and returns a tuple: (opcode, [params])  
 :param data: msg that got from client  
 :return: a tuple: (opcode, [params])  
 """* if len(data) > 1:  
  
 opcode = data[:2]  
 data = data[2:]  
  
 parsed = data.split(',')  
  
 return opcode, parsed  
  
  
def pack\_client\_type(client\_type: str):  
 *"""  
 builds message by protocol and returns packed str  
 :param client\_type: the client type  
 :return: packed str  
 """* return f'35{client\_type}'  
  
  
def pack\_file\_tree(path: str):  
  
 *"""  
 builds message by protocol and returns packed str  
 :param path: location  
 :return: packed str  
 """* opcode = '20'  
 return f'{opcode}{FileHandler.get\_path\_tree(path)}'  
  
  
def pack\_mac(mac: str):  
 *"""  
 builds message by protocol and returns packed str  
 :param mac: mac address  
 :return: packed str  
 """* opcode = '29'  
 return f'{opcode}{mac}'  
  
  
def pack\_status\_create(status: bool, location: str, typ: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param location: location of creation  
 :param typ: type of creation  
 :return: packed str  
 """* packed = '33'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{location},{typ}'  
  
  
def pack\_status\_rename(status: bool, location: str, new\_name: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param location: location of renaming  
 :param new\_name: new name of file\folder  
 :return: packed str  
 """* packed = '22'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{location},{new\_name}'  
  
  
def pack\_status\_delete(status: bool, location: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param location: location of delete  
 :return: packed str  
 """* packed = '24'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{location}'  
  
  
def pack\_status\_move\_to\_server(status: bool, old\_path: str, new\_path: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param old\_path: the old path of the file  
 :param new\_path: the new path of the file  
 :return: packed str  
 """* packed = '26'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{old\_path},{new\_path}'  
  
def pack\_do\_move(new\_path: str, old\_path: str):  
 *"""  
 builds by protocol to move file  
 :param new\_path: the path of the file to save it with  
 :param old\_path: the old path of the file  
 :return: packed str  
 """* packed = '38'  
 return f'{packed}{new\_path},{old\_path}'  
  
  
def pack\_status\_move\_to\_client(status: bool, moved\_to: str, moved\_from: str):  
 *"""  
 packs if worked moving file to client  
 :param status: boolean (success of failure)  
 :param moved\_to: the new path of the file  
 :param moved\_from: the old path of the file  
 :return: packed str  
 """* packed = '39'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{moved\_to},{moved\_from}'  
  
  
def pack\_do\_clone(new\_path: str, old\_path: str):  
 *"""  
 builds by protocol to clone file  
 :param new\_path: the path of the file to save it with  
 :param old\_path: the old path of the file  
 :return: packed str  
 """* packed = '36'  
 return f'{packed}{new\_path},{old\_path}'  
  
  
def pack\_status\_clone\_to\_client(status: bool, cloned\_to: str, cloned\_from: str):  
 *"""  
 packs if worked cloning file to client  
 :param status: boolean (success of failure)  
 :param cloned\_to: the new path of the file  
 :param cloned\_from: the old path of the file  
 :return: packed str  
 """* packed = '37'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{cloned\_to},{cloned\_from}'  
  
def pack\_status\_clone\_to\_server(status: bool, copy\_from: str, copy\_to: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param copy\_to: folder to copy to  
 :param copy\_from: path to copy from  
 :param status: boolean (success of failure)  
 :return: packed str  
 """* packed = '28'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{copy\_from},{copy\_to}'  
  
  
def pack\_status\_open\_file(status: bool, path=''):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param path: path of file if able to open file  
 :param status: boolean (success of failure)  
 :return: packed str  
 """* packed = '17'  
 if status:  
 packed += f'ok,{path}'  
 else:  
 packed += 'no'  
  
 return packed  
  
  
def pack\_status\_change\_file(status: bool, path: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param path: path of the file changed  
 :return: packed str  
 """* packed = '19'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{path}'  
  
  
def pack\_key(key):  
 *"""  
 builds by protocol to send symmetrical key  
 :param key: a symmetrical key  
 :return: packed str  
 """* return b'99'+key

**server\_protocol.py:**

import ast  
  
def unpack(data: str):  
 *"""  
 parsing by protocol and returns a tuple: (opcode, [params])  
 :param data: msg that got from client  
 :return: a tuple: (opcode, [params])  
 """* if len(data) > 1:  
 opcode = data[:2]  
 data = data[2:]  
  
 *# if got file tree* if opcode == '20':  
 parsed = [string\_to\_dict(data)]  
 else:  
 parsed = data.split(',')  
  
 return opcode, parsed  
  
  
  
def string\_to\_dict(data: str):  
 *"""  
 converts a string from protocol to a dictionary  
 :param data: a packed string from protocol  
 :return: a dictionary  
 """* folders = {}  
 lines = data.split('\n')[:-1]  
 for line in lines:  
 sep\_line = line.split('?')  
 directory = sep\_line[0]  
 try:  
 dirs = ast.literal\_eval(sep\_line[1])  
 files = ast.literal\_eval(sep\_line[2])  
 except Exception as e:  
 print(f'in unpacking file tree: {str(e)}')  
 break  
 dirs.append(',')  
 dirs.extend(files)  
 folders[directory] = dirs  
  
 return folders  
  
  
def pack\_status\_mac(status: bool):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :return: packed str  
 """* packed = '34'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return packed  
  
  
def pack\_status\_register(status: bool):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :return: packed str  
 """* packed = '02'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return packed  
  
  
def pack\_status\_login(status: bool):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :return: packed str  
 """* packed = '04'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return packed  
  
  
def pack\_status\_create(status: bool, path: str, typ: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param path: path of creation  
 :param typ: type of creation  
 :return: packed str  
 """* packed = '07'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{path},{typ}'  
  
  
def pack\_status\_rename(status: bool, path: str, new\_name: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param path: path of file  
 :param new\_name: the new name of the file  
 :return: packed str  
 """* packed = '09'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{path},{new\_name}'  
  
  
def pack\_status\_delete(status: bool, location: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param location: location of delete  
 :return: packed str  
 """* packed = '11'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{location}'  
  
  
def pack\_status\_move(status: bool, old\_path: str, new\_path: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param status: boolean (success of failure)  
 :param old\_path: the old path of the file  
 :param new\_path: new path of the file  
 :return: packed str  
 """* packed = '13'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{old\_path},{new\_path}'  
  
  
def pack\_status\_clone(status: bool, copy\_from: str, copy\_to: str):  
 *"""  
 gets a boolean that tells if action worked or not and returns the packed str  
 :param copy\_to: folder to copy to  
 :param copy\_from: path of file to copy  
 :param status: boolean (success of failure)  
 :return: packed str  
 """* packed = '15'  
 if status:  
 packed += 'ok'  
 else:  
 packed += 'no'  
  
 return f'{packed},{copy\_from},{copy\_to}'  
  
  
def pack\_do\_remove(folder: str):  
 *"""  
 gets a certain folder and returns packed by protocol that tells to remove it  
 :param folder: str  
 :return: packed str  
 """* return f'30{folder}'  
  
  
def pack\_ask\_file\_Tree(folder: str):  
 *"""  
 :return: protocol for asking file system  
 """* return f'31{folder}'  
  
  
def pack\_send\_file\_tree(file\_tree: dict):  
  
 *"""  
 builds message by protocol and returns packed str  
 :param file\_tree: the file tree  
 :return: packed str  
 """* return f'05{dict\_to\_string(file\_tree)}'  
  
  
def dict\_to\_string(data: dict):  
 *"""  
 turns a dictionary to string by protocol  
 :param data: the dictionary  
 :return: str  
 """* string = ''  
 for folder in data:  
 string += folder + '?'  
 folder\_content = data[folder]  
 separator\_index = folder\_content.index(',')  
 folder\_list = folder\_content[:separator\_index]  
 folder\_files = folder\_content[separator\_index + 1:]  
 string += f'{folder\_list}?{folder\_files}\n'  
  
 return string  
  
  
def pack\_do\_rename(location: str, new\_name: str):  
 *"""  
 gets location of file/folder and name to rename to, returns packed str that does it by protocol  
 :param location: location of file/folder  
 :param new\_name: location of file/folder  
 :return: packed str  
 """* opcode = '21'  
 return f'{opcode}{location},{new\_name}'  
  
  
def pack\_do\_create(location: str, typ: str):  
 *"""  
 gets location of file/folder and type to create, returns packed str that does it by protocol  
 :param location: location of file/folder  
 :param typ: type of file or folder  
 :return: packed str  
 """* opcode = '32'  
 return f'{opcode}{location},{typ}'  
  
def pack\_do\_delete(location: str):  
 *"""  
 gets location of file/folder to delete, returns packed str that does it by protocol  
 :param location: location of file/folder  
 :return: packed str  
 """* opcode = '23'  
 return f'{opcode}{location}'  
  
  
def pack\_do\_move(old\_location: str, new\_location: str):  
 *"""  
 gets location of file to move and new location to move it, returns packed str that does it by protocol  
 :param old\_location: location of file  
 :param new\_location: new location of file  
 :return: packed str  
 """* opcode = '25'  
 return f'{opcode}{old\_location},{new\_location}'  
  
  
def pack\_do\_clone(old\_location: str, new\_location: str):  
 *"""  
 gets location of file to clone and new location to clone it, returns packed str that does it by protocol  
 :param old\_location: location of file  
 :param new\_location: new location of file  
 :return: packed str  
 """* opcode = '27'  
 return f'{opcode}{old\_location},{new\_location}'  
  
  
def pack\_public\_key(public\_key):  
 *"""  
 get public key and packs it by protocol  
 :param public\_key: a public key  
 :return: packed str  
 """* return b'98'+public\_key

**user\_client\_protocol.py:**

import ast  
  
def unpack(data: str):  
 *"""  
 parsing by protocol and returns a tuple: (opcode, [params])  
 :param data: msg that got from client  
 :return: a tuple: (opcode, [params])  
 """* if len(data) > 1:  
  
 opcode = data[:2]  
 data = data[2:]  
  
 *# if got file tree* if opcode == '05':  
 folders = string\_to\_dict(data)  
 parsed = [folders]  
 else:  
 parsed = data.split(',')  
  
 return opcode, parsed  
  
  
def string\_to\_dict(data: str):  
 *"""  
 converts a string from protocol to a dictionary  
 :param data: a packed string from protocol  
 :return: a dictionary  
 """* folders = {}  
 lines = data.split('\n')[:-1]  
 for line in lines:  
 sep\_line = line.split('?')  
 directory = sep\_line[0]  
 try:  
 dirs = ast.literal\_eval(sep\_line[1])  
 files = ast.literal\_eval(sep\_line[2])  
 except Exception as e:  
 print(f'in unpacking file tree: {str(e)}')  
 break  
 dirs.append(',')  
 dirs.extend(files)  
 folders[directory] = dirs  
  
 return folders  
  
def pack\_register(username: str, password: str):  
 *"""  
 builds message by protocol and returns packed str  
 :param username: str  
 :param password: str  
 :return: packed str  
 """* opcode = '01'  
 return f'{opcode}{username},{password}'  
  
  
def pack\_sign\_in(username: str, password: str, mac: str):  
 *"""  
 builds message by protocol and returns packed str  
 :param username: str  
 :param password: str  
 :param mac: str  
 :return: packed str  
 """* opcode = '03'  
 return f'{opcode}{username},{password},{mac}'  
  
  
def pack\_do\_create(location: str, type: str):  
 *"""  
 builds message by protocol and return packed str  
 :param location: str  
 :param type: str  
 :return: packed str  
 """* opcode = '06'  
 return f'{opcode}{location},{type}'  
  
  
def pack\_do\_rename(location: str, new\_name: str):  
 *"""  
 builds message by protocol and return packed str  
 :param location: str  
 :param new\_name: str  
 :return: packed str  
 """* opcode = '08'  
 return f'{opcode}{location},{new\_name}'  
  
  
def pack\_do\_delete(location: str):  
 *"""  
 builds message by protocol and return packed str  
 :param location: str  
 :return: packed str  
 """* opcode = '10'  
 return f'{opcode}{location}'  
  
  
def pack\_do\_move(old\_location: str, new\_location: str):  
 *"""  
 builds message by protocol and return packed str  
 :param old\_location: str  
 :param new\_location: str  
 :return: packed str  
 """* opcode = '12'  
 return f'{opcode}{old\_location},{new\_location}'  
  
  
def pack\_do\_clone(old\_location: str, new\_location: str):  
 *"""  
 builds message by protocol and return packed str  
 :param old\_location: str  
 :param new\_location: str  
 :return: packed str  
 """* opcode = '14'  
 return f'{opcode}{old\_location},{new\_location}'  
  
  
def pack\_do\_open\_file(location: str):  
 *"""  
 builds message by protocol and return packed str  
 :param location: str  
 :return: packed str  
 """* opcode = '16'  
 return f'{opcode}{location}'  
  
  
def pack\_change\_file(location: str):  
 *"""  
 builds message by protocol and return packed str  
 :param data\_len: len of file  
 :param location: str  
 :return: packed str  
 """* opcode = '18'  
 return f'{opcode}{location}'

**file\_handler.py:**

import ctypes  
import os  
from Reflection.settings import Settings  
import shutil  
from subprocess import Popen  
  
class FileHandler:  
 root = Settings.root  
  
 def \_\_init\_\_(self, username: str):  
 self.my\_ip = Settings.get\_ip()  
 self.username = username  
 self.user\_path = self.root + username + '\\'  
  
 def create\_root(self):  
 *"""  
 creates root directory hidden  
 :return:  
 """* FileHandler.\_create\_hidden\_dir(self.root)  
 FileHandler.create(self.user\_path, 'fld')  
  
 def is\_local(self, path):  
 *"""  
 gets path and returns if local  
 :param path: path of dir  
 :return: if path is local or not  
 """* return path.startswith(self.user\_path + self.my\_ip) and os.path.exists(FileHandler.remove\_ip(self.username, path))  
  
  
 @ staticmethod  
 def split\_name\_typ(name: str):  
 *"""  
 splits name and type  
 :param name: name of file  
 :return: name parted (name, type)  
 """* typ\_index = name.rfind('.')  
 typ = name[typ\_index + 1:]  
 name = name[:typ\_index]  
 return name, typ  
  
 @ staticmethod  
 def build\_name\_for\_file(file\_tree: dict, new\_dir\_path: str, path\_of\_file: str, adding: str):  
 *"""  
 building a name to a file to fit to directory  
 :param file\_tree: the file tree using  
 :param new\_dir\_path: path of new folder  
 :param path\_of\_file: path of a file to build name for  
 :param adding: adding at end of file for new name  
 :return: new name of the file  
 """* \_, file\_name = FileHandler.split\_path\_last\_part(path\_of\_file)  
 new\_file\_name = file\_name  
 if file\_name in file\_tree[new\_dir\_path]:  
 if adding not in file\_name:  
 parted\_file\_name = file\_name.split('.')  
 parted\_file\_name[0] += adding  
 new\_file\_name = '.'.join(parted\_file\_name)  
  
 *# adding count numbers if copied several times* count = 1  
 while True:  
 if new\_file\_name in file\_tree[new\_dir\_path]:  
 parted\_file\_name = new\_file\_name.split('.')  
 parted\_file\_name[0] = parted\_file\_name[0].split('-')[0] + f'-{count}'  
 new\_file\_name = '.'.join(parted\_file\_name)  
 count += 1  
 else:  
 break  
  
 return new\_file\_name  
  
 @ staticmethod  
 def direct\_copy\_file(from\_path: str, to: str):  
 *"""  
 copies a file  
 :param from\_path: original path  
 :param to: new path  
 :return: if was able to copy file  
 """* worked = os.path.isfile(from\_path)  
 print('file\_path:', from\_path)  
 if worked:  
 shutil.copy(from\_path, to)  
 return worked  
  
 @ staticmethod  
 def move(old\_path: str, new\_path: str):  
 *"""  
 moves a file  
 :param old\_path: old path of file  
 :param new\_path: the new path of the file  
 :return: if success in moving file  
 """* worked = os.path.isfile(old\_path)  
 if worked:  
 shutil.move(old\_path, new\_path)  
 return worked  
  
  
  
 @ staticmethod  
 def rename(location: str, new\_name: str):  
 *"""  
 gets path and renames it  
 :param location: path to rename  
 :param new\_name: new name to rename to  
 :return: if renamed or not  
 """* rename = False  
 new\_path, \_ = FileHandler.split\_path\_last\_part(location)  
 new\_path += f'\\{new\_name}'  
 print('new\_path:', new\_path)  
 if os.path.exists(location) and not os.path.exists(new\_path):  
 try:  
 os.rename(location, new\_path)  
 except Exception:  
 rename = False  
 else:  
 rename = True  
  
 return rename  
  
  
 @staticmethod  
 def delete(location: str):  
 *"""  
 gets path and deletes it from computer  
 :param location: path to delete  
 """* delete = False  
 if os.path.isdir(location):  
 delete = True  
 shutil.rmtree(location)  
 elif os.path.isfile(location):  
 delete = True  
 os.remove(location)  
  
 return delete  
  
 @staticmethod  
 def split\_path\_last\_part(path):  
 *"""  
 gets path and removes last part of it  
 :param path: a path  
 :return: path updated, last part  
 """  
 # Split the path into its components* path\_components = path.split(os.path.sep)  
  
 *# Get the last component* last\_part = path\_components[-1]  
  
 *# Remove the last component* updated\_path\_components = path\_components[:-1]  
  
 *# Join the updated components back into a path* updated\_path = os.path.sep.join(updated\_path\_components)  
  
 return updated\_path, last\_part  
  
 @staticmethod  
 def open\_file(path: str):  
 *"""  
 gets path and opens file  
 :param path: path of file  
 """* Popen(['start', path], shell=True)  
  
  
 @staticmethod  
 def insert\_ip(path: str, username: str, ip: str):  
 *"""  
 gets username and path and inserts ip in it  
 :param username: username  
 :param path: path to insert into  
 :param ip: ip to insert  
 :return: path with ip in it  
 """* ip\_to\_insert = '\\' + ip  
 return path.replace(username, username + ip\_to\_insert, 1)  
  
  
 @staticmethod  
 def extract\_ip(username: str, path: str):  
 *"""  
 gets path and extracts ip out of it  
 :param path: path  
 :param username: the username inside the path  
 :return: ip in path  
 """* ip = None  
 if username in path:  
 user\_path\_len = path.index(username) + len(username) + 1  
 if user\_path\_len < len(path):  
 ip = path[user\_path\_len:]  
 if '\\' in ip:  
 ip = ip[:ip.index('\\')]  
  
 return ip  
  
 @staticmethod  
 def remove\_ip(username: str, path: str):  
 *"""  
 gets path and removes ip out of it  
 :param username: username of user  
 :param path: path  
 :return: path without ip  
 """* if username in path:  
 user\_path\_len = path.index(username) + len(username) + 1  
 if user\_path\_len < len(path):  
 ip = path[user\_path\_len:]  
  
 if '\\' in ip:  
 ip = '\\' + ip[:ip.index('\\')]  
  
 if ip in path:  
 print('hi')  
 path = path.replace(ip, '', 1)  
  
 return path  
  
 @staticmethod  
 def create(path: str, typ: str):  
 *"""  
 gets path to create, creates it if does not exist  
 :param path: path to dir  
 :param typ: type of creation  
 :return: if created or not  
 """* created = True  
 if not os.path.exists(path):  
 if typ == 'fld':  
 os.makedirs(path)  
 else:  
 with open(f'{path}.{typ}', 'w') as fp:  
 pass  
 else:  
 created = False  
  
 return created  
  
 @staticmethod  
 def \_create\_hidden\_dir(path: str):  
 *"""  
 gets path for dir and creates it hidden  
 :param path: path for dir  
 :return: None  
 """* FileHandler.create(path, 'fld')  
 attrs = os.stat(path).st\_file\_attributes  
 if not attrs & 2 == 2:  
 *# Get the file attributes* attrs = ctypes.windll.kernel32.GetFileAttributesW(path)  
 *# Add the 'hidden' attribute* ctypes.windll.kernel32.SetFileAttributesW(path, attrs | 2)  
  
 @staticmethod  
 def get\_user(path: str):  
 *"""  
 gets path and returns user inside path  
 :param path: a path  
 :return: user inside path  
 """* path = path.split(os.path.sep)  
 user = None  
 if len(path) >= 3:  
 user = path[2]  
 return user  
  
 @staticmethod  
 def get\_path\_tree(path: str):  
 *"""  
 returns path to dir returns tree of dir  
 :param path: path to dir  
 :return: file tree of path  
 """* file\_tree = ''  
 for part in os.walk(path):  
 part = f'{part[0]}?{part[1]}?{part[2]}\n'  
 file\_tree += part  
  
 return file\_tree

**process\_handler.py:**

import psutil  
  
image\_types = ["apng", "avif", "gif", "jpg", "jpeg", "jfif", "pjpeg", "pjp", "png", "svg", "webp", "bmp", "ico",  
 "cur",  
 "tif", "tiff"]  
  
default\_for\_type = {'docx': 'WINWORD.EXE',  
 'pptx': 'POWERPNT.EXE',  
 'ppt': 'POWERPNT.EXE',  
 \*\*{img: 'PhotosService.exe' for img in image\_types},  
 'xlsx': 'EXCEL.EXE'}  
  
def get\_all\_pid(process\_name):  
 *"""  
 gets all processes running with process\_name got  
 :param process\_name: the type of process  
 :return: all processes running with the same process name  
 """* current = []  
 for proc in psutil.process\_iter():  
 if proc.name() == process\_name:  
 current.append(proc.pid)  
  
 return current  
  
def get\_process\_name(file\_path: str):  
 *"""  
 gets process name by file path  
 :param file\_path: path of file  
 :return: process name of file  
 """* file\_extension = file\_path[file\_path.rfind('.') + 1:]  
 if file\_extension in default\_for\_type:  
 process\_name = default\_for\_type[file\_extension]  
 else:  
 process\_name = 'Notepad.exe'  
  
 return process\_name  
  
def wait\_for\_process\_to\_close(pid):  
 *"""  
 waiting for process to close  
 :param pid: the process id  
 :return: None  
 """* while psutil.pid\_exists(pid):  
 pass

**Asymmetric\_ecnryption.py**

from Crypto.Cipher import PKCS1\_v1\_5  
from Crypto.PublicKey import RSA  
  
  
class AsymmetricEncryption:  
  
 def \_\_init\_\_(self, private\_key=None, public\_key=None):  
  
 if not private\_key and not public\_key:  
 key = RSA.generate(2048)  
 self.public\_key = key.publickey().export\_key()  
 self.private\_key = key.export\_key()  
  
 elif not private\_key or not public\_key:  
 self.private\_key = private\_key  
 self.public\_key = public\_key  
 else:  
 raise ValueError('Either provide two arguments or none')  
  
 @staticmethod  
 def encrypt\_msg(msg, receiver\_key):  
 *"""  
 gets a msg and a public key  
 :param msg: msg to encrypt  
 :param receiver\_key: key to encrypt with  
 :return: encrypted msg  
 """* if type(msg).\_\_name\_\_ != 'bytes':  
 msg = msg.encode()  
 cipher = PKCS1\_v1\_5.new(RSA.import\_key(receiver\_key))  
 return cipher.encrypt(msg)  
  
 def decrypt\_msg(self, msg, is\_str=False):  
 *"""  
 gets an encrypted msg and if msg is a string  
 :param msg:  
 :param is\_str:  
 :return:  
 """* cipher = PKCS1\_v1\_5.new(RSA.import\_key(self.private\_key))  
 decrypted\_msg = cipher.decrypt(msg, None)  
 if is\_str:  
 decrypted\_msg = decrypted\_msg.decode()  
  
 return decrypted\_msg  
  
  
 def get\_public\_key(self):  
 *"""  
 returns public key  
 :return: public key  
 """* return self.public\_key  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
  
 encryption = AsymmetricEncryption()  
 msg = 'hello' *# limit 245* msg\_encrypted = encryption.encrypt\_msg(msg, encryption.get\_public\_key())  
 print('len', len(msg\_encrypted))  
 msg = encryption.decrypt\_msg(msg\_encrypted)  
 print(msg)

**symmetrical\_encryption.py:**

import base64  
import hashlib  
  
import Crypto.Util.Padding  
from Crypto import Random  
from Crypto.Cipher import AES  
from Crypto.Random import get\_random\_bytes  
  
  
class SymmetricalEncryption:  
  
 def \_\_init\_\_(self, key=None):  
  
 if not key:  
 self.key = get\_random\_bytes(32)  
 else:  
 self.key = key  
 self.encryption = hashlib.sha256(self.key).digest()  
  
 def encrypt(self, raw):  
 *"""  
 gets data and returns it encrypted  
 :param raw: data  
 :return: encrypted data  
 """* raw = self.\_pad(raw)  
 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, is\_image=False):  
 *"""  
 gets encrypted data and returns it decrypted  
 :param enc: data encrypted  
 :param is\_image: if the data is image or not  
 :return: data decrypted  
 """* enc = base64.b64decode(enc)  
 iv = enc[:AES.block\_size]  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
 decrypted\_msg = self.\_unpad(cipher.decrypt(enc[AES.block\_size:]))  
 if not is\_image:  
 decrypted\_msg = decrypted\_msg.decode('utf-8')  
  
 return decrypted\_msg  
  
 def \_pad(self, s):  
 *"""  
 gets data and pads it  
 :param s: data  
 :return: data with pad  
 """* if type(s).\_\_name\_\_ == 'str':  
 s = s.encode()  
 return Crypto.Util.Padding.pad(s, 16, style='pkcs7')  
  
 @staticmethod  
 def \_unpad(s):  
 *"""  
 gets data pad returns it without pad  
 :param s: data  
 :return: data without pad  
 """* return Crypto.Util.Padding.unpad(s, 16, style='pkcs7')  
  
 @staticmethod  
 def hash(data):  
 *"""  
 gets data and hashes it  
 :param data: data  
 :return: data hashed  
 """* data = bytes(str(data), 'utf-8')  
 return hashlib.sha256(data).digest()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 encryption = SymmetricalEncryption()  
 data = encryption.encrypt('hello')  
 print(data)  
 data = encryption.decrypt(data)  
 print(data)

**db.py:**

import sqlite3  
  
from Reflection.encryption import symmetrical\_encryption  
  
  
class Db:  
 def \_\_init\_\_(self):  
 self.db\_name = "reflection\_db"  
 self.conn = None  
 self.curr = None  
 self.users\_tbl = "user\_pass"  
 self.mac\_tbl = "user\_mac"  
 self.encryption = symmetrical\_encryption.SymmetricalEncryption()  
 self.\_create\_db()  
  
 def \_create\_db(self):  
 *"""  
 connect to DB and create tables if they do not exist  
 :return:  
 """* self.conn = sqlite3.connect(self.db\_name)  
 self.curr = self.conn.cursor()  
 sql = ["CREATE TABLE IF NOT EXISTS " + self.users\_tbl +  
 "(user VARCHAR(30), password VARCHAR(30), PRIMARY KEY(user))",  
 "CREATE TABLE IF NOT EXISTS " + self.mac\_tbl +  
 "(user VARCHAR(30), mac VARCHAR(17), PRIMARY KEY(user, mac))"]  
  
 [self.curr.execute(s) for s in sql]  
 self.conn.commit()  
  
 def \_username\_exists(self, username: str) -> bool:  
 *"""  
 check if username exists in the user\_pass table  
 :param username: username  
 :return: boolean  
 """* sql = "SELECT user FROM " + self.users\_tbl + " WHERE user = ?"  
 self.curr.execute(sql, (username,))  
 return self.curr.fetchone() is not None  
  
 def \_username\_mac\_exists(self, username: str, mac: str) -> bool:  
 *"""  
 check if username and mac together exists in the user\_mac table  
 :param username: username  
 :param mac: mac address  
 :return: boolean  
 """* sql = "SELECT user FROM " + self.mac\_tbl + " WHERE user = ? and mac = ?"  
 self.curr.execute(sql, (username, mac,))  
 return self.curr.fetchone() is not None  
  
  
 def add\_user(self, username: str, password: str) -> bool:  
 *"""  
 adds user to user\_pass table  
 :param username: username  
 :param password: password  
 :return: True or False  
 """* flag = True  
 if self.\_username\_exists(username):  
 flag = False  
 else:  
 print(self.users\_tbl)  
 sql = "INSERT INTO " + self.users\_tbl + " VALUES(?,?)"  
 self.curr.execute(sql, (username, self.encryption.hash(password),))  
 self.conn.commit()  
  
 return flag  
  
  
 def add\_user\_mac(self, username: str, mac: str) -> bool:  
 *"""  
 adds user to user\_pass table  
 :param username: username  
 :param mac: mac address  
 :return: True or False  
 """* flag = True  
 if self.\_username\_mac\_exists(username, mac):  
 flag = False  
 else:  
 sql = "INSERT INTO " + self.mac\_tbl + " VALUES(?,?)"  
 self.curr.execute(sql, (username, mac,))  
 self.conn.commit()  
  
 return flag  
  
  
 def change\_password(self, username: str, password: str):  
 *"""  
 changing password of username and returns a boolean  
 :param username: username  
 :param password: password  
 :return: a boolean  
 """* flag = True  
 if not self.\_username\_exists(username):  
 flag = False  
  
 else:  
 sql = "UPDATE " + self.users\_tbl\  
 + " SET password = ? WHERE user = ?"  
 self.curr.execute(sql, (self.encryption.hash(password), username))  
 self.conn.commit()  
 return flag  
  
  
 def get\_password(self, username: str):  
 *"""  
 return the password of a username if exists  
 :param username: username  
 :return: password or None  
 """* if self.\_username\_exists(username):  
 sql = "SELECT password FROM " + self.users\_tbl + " WHERE user = ?"  
 self.curr.execute(sql, (username,))  
 password = self.curr.fetchone()[0]  
 return password  
  
 def get\_macs(self, username):  
 *"""  
 gets username and returns a list of all macs related to that user  
 :param username: username  
 :return: a list of all macs related to user  
 """* sql = "SELECT mac FROM " + self.mac\_tbl + " WHERE user = ?"  
 self.curr.execute(sql, (username,))  
 macs = self.curr.fetchall()  
 return [row[0] for row in macs]  
  
  
 def get\_users(self, mac):  
 *"""  
 gets username and returns a list of all macs related to that user  
 :param mac: mac address  
 :return: a list of all macs related to user  
 """* sql = "SELECT user FROM " + self.mac\_tbl + " WHERE mac = ?"  
 self.curr.execute(sql, (mac,))  
 users = self.curr.fetchall()  
 return [row[0] for row in users]  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 db = Db()  
 print(db.add\_user('yotam\_king', '1234'))  
 print(db.get\_users('m13414'))

**client\_comm.py:**

import queue  
import socket  
import sys  
import threading  
import time  
import os  
  
  
from Reflection.encryption import symmetrical\_encryption  
from Reflection.protocols import general\_client\_protocol  
from Reflection.protocols import user\_client\_protocol  
from Reflection.local\_handler.file\_handler import FileHandler  
from Reflection.settings import Settings  
from Reflection.encryption.asymmetric\_encryption import AsymmetricEncryption  
  
  
  
class ClientComm:  
  
 main\_server\_port = Settings.server\_port  
 file\_receive\_opcodes = ('17', '18')  
 def \_\_init\_\_(self, server\_ip, port, rcv\_q, send\_len, client\_type):  
 self.send\_len = send\_len  
 self.server\_ip = server\_ip  
 self.port = port  
 self.rcv\_q = rcv\_q  
 self.client\_type = client\_type  
 self.server = socket.socket()  
 self.running = True  
 self.server.settimeout(10)  
  
 self.symmetric = None  
  
 threading.Thread(target=self.\_main\_loop, daemon=True).start()  
 time.sleep(0.3)  
  
 def \_main\_loop(self):  
 *"""  
 receives  
 :return:  
 """* try:  
 self.server.connect((self.server\_ip, self.port))  
 except Exception as e:  
 self.close()  
 sys.exit("server is down, try again later")  
  
 self.\_key\_exchange()  
 while True:  
 try:  
 length = int(self.server.recv(self.send\_len).decode())  
 data = self.server.recv(length)  
 except socket.timeout as e:  
 continue  
 except Exception as e:  
 print(e)  
 self.close()  
 sys.exit('server is no good')  
 else:  
 data = self.symmetric.decrypt(data)  
 if data[:2] in self.file\_receive\_opcodes:  
 threading.Thread(target=self.\_receive\_file(data))  
 else:  
 self.rcv\_q.put(data)  
  
 def \_receive\_file(self, header):  
 *"""  
 gets header of file, receives file  
 :param header: header of file  
 :return: None  
 """* if header[2:] == 'no':  
 self.rcv\_q.put(header)  
 return  
 try:  
 data\_len = int(self.server.recv(self.send\_len).decode())  
 except Exception as e:  
 self.close()  
 print('error in receiving file:', str(e))  
  
 else:  
 file\_is\_ok = True  
 file = bytearray()  
  
 *# receiving file in parts* while len(file) < data\_len and file\_is\_ok:  
  
 size = data\_len - len(file)  
 if size > 1024:  
 try:  
 file.extend(self.server.recv(1024))  
 except Exception as e:  
 print('error in receiving file:', str(e))  
 file\_is\_ok = False  
  
 else:  
 try:  
 file.extend(self.server.recv(size))  
 except Exception as e:  
 print('error in receiving file:', str(e))  
 file\_is\_ok = False  
  
 if file\_is\_ok:  
 file = bytes(file)  
 file = self.symmetric.decrypt(file, True)  
  
 path = header.split(',')[1]  
 path, name = FileHandler.split\_path\_last\_part(path)  
 if FileHandler.root in path:  
 path = path.replace(FileHandler.root, Settings.tmp\_directory\_path, 1)  
 *# creating folder for file* FileHandler.create(path, 'fld')  
  
 *# creating file* path += '\\' + name  
 with open(path, 'wb') as save:  
 save.write(file)  
  
 self.rcv\_q.put(general\_client\_protocol.pack\_status\_open\_file(True, path))  
  
  
 def \_key\_exchange(self):  
 *"""  
 gets server's public key and sends server symmetric key  
 :return: None  
 """* try:  
 length = int(self.server.recv(self.send\_len).decode())  
 data = self.server.recv(length)  
  
 except Exception as e:  
 self.close()  
 sys.exit("server is down, try again later")  
  
 public\_key = data[2:]  
 print('pulic key:', public\_key)  
 self.symmetric = symmetrical\_encryption.SymmetricalEncryption()  
 key\_to\_send = general\_client\_protocol.pack\_key(self.symmetric.key)  
 key\_to\_send = AsymmetricEncryption.encrypt\_msg(key\_to\_send, public\_key)  
 try:  
 self.server.send(str(len(key\_to\_send)).zfill(self.send\_len).encode())  
 self.server.send(key\_to\_send)  
 except Exception as e:  
 self.close()  
 sys.exit('could not send server key encrypted')  
  
 else:  
 self.\_send\_client\_type()  
  
  
  
 def \_send\_client\_type(self):  
 *"""  
 sends server client type  
 """* self.send(general\_client\_protocol.pack\_client\_type(self.client\_type))  
  
  
 def send(self, data):  
 *"""  
 sending data to server  
 :param data: data to send  
 :return: None  
 """* print(f'sending: {data}')  
 if self.symmetric:  
 data = self.symmetric.encrypt(data)  
  
 try:  
 print(f'data to send: {str(len(data)).zfill(self.send\_len).encode() + data}')  
 self.server.send(str(len(data)).zfill(self.send\_len).encode() + data)  
 except Exception as e:  
 print('client comm - send:', str(e))  
 self.close()  
 sys.exit("server is down, try again later")  
  
 def send\_file(self, header, data):  
 *"""  
 sends file to server  
 :param header: header of the file  
 :param data: the data of the file (binary)  
 :return: None  
 """* header = self.symmetric.encrypt(header)  
 data = self.symmetric.encrypt(data)  
 header\_len = str(len(header)).zfill(self.send\_len).encode()  
 data\_len = str(len(data)).zfill(self.send\_len).encode()  
 try:  
 self.server.send(header\_len + header + data\_len + data)  
 except Exception as e:  
 self.close()  
 print('client comm - send file', str(e))  
 sys.exit("server is down, try again later")  
  
 def close(self):  
 self.server.close()  
 self.rcv\_q.put('00')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 rcv\_q = queue.Queue()  
 server = ClientComm('127.0.0.1', 2500, rcv\_q, 8, 'G')  
 server.send('05hi')  
 while True:  
 if not rcv\_q.empty():  
 data = rcv\_q.get()  
 data = user\_client\_protocol.unpack(data)  
 os.system('start "" "' + data[0] + '"')

**server\_comm.py:**

import queue  
import socket  
import threading  
  
import select  
from Reflection.encryption import asymmetric\_encryption  
from Reflection.encryption import symmetrical\_encryption  
from Reflection.protocols import general\_client\_protocol  
from Reflection.protocols import server\_protocol  
from Reflection.settings import Settings  
from Reflection.local\_handler.file\_handler import FileHandler  
  
  
class ServerComm:  
  
 main\_server\_port = Settings.server\_port  
 file\_receive\_opcodes = ('17', '18', '36', '38')  
 def \_\_init\_\_(self, port, rcv\_q, send\_len):  
  
 self.port = port  
 self.rcv\_q = rcv\_q  
 self.send\_len = send\_len  
 self.socket = socket.socket()  
 *# socket: ip* self.open\_clients = {}  
 self.a\_encrypt = asymmetric\_encryption.AsymmetricEncryption()  
 self.is\_running = False  
 self.receiving\_files = []  
 self.my\_ip = Settings.get\_ip()  
 threading.Thread(target=self.\_main\_loop).start()  
  
 def \_main\_loop(self):  
 *"""  
 waiting for clients to connect and puts in queue messages got from clients  
 """* 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)  
 *# new client* for current\_socket in rlist:  
 if current\_socket is self.socket:  
  
 client, addr = self.socket.accept()  
 print(f'{addr[0]} connected!')  
 threading.Thread(target=self.\_key\_exchange, args=(client, addr[0])).start()  
 continue  
  
 *# exist client* if current\_socket in self.receiving\_files:  
 continue  
 try:  
 length = int(current\_socket.recv(self.send\_len).decode())  
 data = current\_socket.recv(length)  
 except Exception as e:  
 print("main server in server comm: " + str(e))  
 self.disconnect\_client(current\_socket)  
 else:  
 *# exchanged keys already* if current\_socket in self.open\_clients:  
 encryption = self.open\_clients[current\_socket][1]  
 data = encryption.decrypt(data)  
  
 if data[:2] in self.file\_receive\_opcodes:  
 self.receiving\_files.append(current\_socket)  
  
 *# maybe thread  
 # self.\_receive\_file(current\_socket, data)* threading.Thread(target=self.\_receive\_file, args=(current\_socket, data, )).start()  
  
 else:  
 print(f'data: {data}')  
 self.rcv\_q.put((self.open\_clients[current\_socket][0], data))  
  
  
 def \_receive\_file(self, client, header):  
 *"""  
 gets socket of client and header of file, receives file  
 :param client: socket of client  
 :param header: header of file  
 :return: None  
 """* count = 0  
 try:  
 data\_len = int(client.recv(self.send\_len).decode())  
 except Exception as e:  
 print('error in receiving file:', str(e))  
  
 else:  
 file\_is\_ok = True  
 file = bytearray()  
  
 *# receiving file in parts* while len(file) < data\_len and file\_is\_ok:  
 size = data\_len - len(file)  
 if size > 1024:  
 try:  
 got = client.recv(1024)  
 count += 1  
 file.extend(got)  
 except Exception as e:  
 self.disconnect\_client(client)  
 print('error in receiving file:', str(e))  
 file\_is\_ok = False  
  
 else:  
 try:  
 got = client.recv(size)  
 count += 1  
 file.extend(got)  
 except Exception as e:  
 self.disconnect\_client(client)  
 print('error in receiving file:', str(e))  
 file\_is\_ok = False  
  
 status = 'ok'  
 opcode, header = general\_client\_protocol.unpack(header)  
 path = header[0]  
 if file\_is\_ok:  
 file = bytes(file)  
 file = self.open\_clients[client][1].decrypt(file, True)  
 print(f'saving in {path}: {file}')  
 username = FileHandler.get\_user(path)  
 local\_path = FileHandler.remove\_ip(username, path)  
 try:  
 *# creating file* with open(local\_path, 'wb') as save:  
 save.write(file)  
 except Exception:  
 status = 'no'  
 else:  
 status = 'no'  
  
 client\_ip = self.open\_clients[client][0]  
 header = ','.join(header)  
 self.rcv\_q.put((client\_ip, f'{opcode}{status},{header}'))  
 if client in self.receiving\_files:  
 self.receiving\_files.remove(client)  
  
 def disconnect\_client(self, ip, from\_logic=False):  
 *"""  
 gets client or ip to disconnect informs server and calls \_disconnect\_client  
 :param ip: ip to disconnect  
 :param from\_logic: is called from logic  
 :return: None  
 """* if type(ip).\_\_name\_\_ == 'str' or type(ip).\_\_name\_\_ == 'tuple':  
 client = self.\_find\_socket\_by\_ip(ip)  
 else:  
 client = ip  
 if ip in self.open\_clients:  
 ip = self.open\_clients[client][0]  
  
 if not from\_logic:  
 self.rcv\_q.put((ip, 'close'))  
 self.\_disconnect\_client(client)  
  
 def \_disconnect\_client(self, client):  
 *"""  
 gets ip, deletes ip from all dictionaries and closes it's socket  
 :param client: client socket  
 :return: None  
 """* if client in self.open\_clients:  
 print(f'{self.open\_clients[client][0]} - disconnected')  
 del self.open\_clients[client]  
  
 if client in self.receiving\_files:  
 self.receiving\_files.remove(client)  
  
 if client:  
 print(client)  
 client.close()  
  
 def \_key\_exchange(self, client, ip):  
 *"""  
 exchanging keys with client, puts client in open clients at the end  
 :param client: client socket  
 :param ip: ip address  
 :return:  
 """* self.send(client, server\_protocol.pack\_public\_key(self.a\_encrypt.get\_public\_key()), False)  
 try:  
 length = int(client.recv(self.send\_len).decode())  
 data = client.recv(length)  
 data = self.a\_encrypt.decrypt\_msg(data)  
 except Exception as e:  
 print("error in key\_exchange: " + str(e))  
 self.disconnect\_client(client)  
 else:  
  
 encryption = symmetrical\_encryption.SymmetricalEncryption(data[2:])  
  
 self.\_get\_client\_type(client, ip, encryption)  
  
 def \_get\_client\_type(self, client, ip, encryption):  
 *"""  
 receives client type from client  
 :param client: socket of client  
 :param ip: ip of client  
 :return: None  
 """* try:  
 length = int(client.recv(self.send\_len).decode())  
 data = client.recv(length)  
 print('data:', data)  
 except Exception as e:  
 print("error in key\_exchange: " + str(e))  
 self.disconnect\_client(client)  
 else:  
 data = encryption.decrypt(data)  
 typ = data[-1]  
 if typ != 'G' and typ != 'U':  
 self.disconnect\_client(client)  
 else:  
 self.open\_clients[client] = [(ip, typ)]  
 self.open\_clients[client].append(encryption)  
 print('finished transferring client type')  
  
  
 def \_find\_socket\_by\_ip(self, ip):  
 *"""  
 gets ip and returns the socket matched  
 :param ip: ip got (string)  
 :return: socket matched (socket)  
 """* client = None  
 for soc in self.open\_clients.keys():  
 if self.open\_clients[soc][0] == ip:  
  
 client = soc  
 break  
 return client  
  
 def send(self, ips\_to\_send, data, encrypt=True):  
 *"""  
 sending ip a msg  
 :param ips\_to\_send: the ip to send to and general or user  
 :param data: data to send  
 :param encrypt: to encrypt data or not  
 :return: None  
 """* if type(ips\_to\_send).\_\_name\_\_ != 'list':  
 ips\_to\_send = [ips\_to\_send]  
  
 for ip in ips\_to\_send:  
 if type(ip).\_\_name\_\_ == 'str' or type(ip).\_\_name\_\_ == 'tuple':  
 sock = self.\_find\_socket\_by\_ip(ip)  
 else:  
 sock = ip  
  
 if sock and self.is\_running:  
 print(f'sending to {ip}:', data)  
 if encrypt and sock in self.open\_clients:  
 encryption = self.open\_clients[sock][1]  
 data = encryption.encrypt(data)  
  
 try:  
 sock.send(str(len(data)).zfill(self.send\_len).encode())  
 sock.send(data)  
 except Exception as e:  
 print('server comm - send', str(e))  
 self.disconnect\_client(sock)  
  
 def send\_file(self, ip, header, data):  
 *"""  
 sends file to client  
 :param ip: ip of client  
 :param header: header of the file  
 :param data: data of the file (binary)  
 :return: None  
 """* client = self.\_find\_socket\_by\_ip(ip)  
 if client:  
 symmetric = self.open\_clients[client][1]  
 header = symmetric.encrypt(header)  
 data = symmetric.encrypt(data)  
 header\_len = str(len(header)).zfill(self.send\_len).encode()  
 data\_len = str(len(data)).zfill(self.send\_len).encode()  
 try:  
 client.send(header\_len + header + data\_len + data)  
 except Exception as e:  
 print('server comm - send', str(e))  
 self.disconnect\_client(client)  
  
 def close\_server(self):  
 self.is\_running = False  
  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 msg\_q = queue.Queue()  
 server = ServerComm(2500, msg\_q, 8)  
 file\_name = r'T:\public\cyber\ophir\Reflection\Reflection\comms\cat.jpg'  
 with open(file\_name, 'rb') as f:  
 b = f.read()  
  
  
 while True:  
 if not msg\_q.empty():  
  
 data = msg\_q.get()  
 print(f'sending {data[0]}')  
 server.send(data[0], general\_client\_protocol.pack\_status\_open\_file(True))  
 server.send(data[0], b)

**Graphics.py:**

import time  
import wx  
  
import os  
from pubsub import pub  
from Reflection import settings  
from queue import Queue  
from Reflection.graphics.tree\_graphic import TreeFrame  
class MyFrame(wx.Frame):  
  
 def \_\_init\_\_(self, logic\_q: Queue):  
 super(MyFrame, self).\_\_init\_\_(None, title="", size=(500, 500))  
 *# create status bar* self.CreateStatusBar(1)  
 self.SetStatusText("Developed by ophir")  
 *# create main panel - to put on the others panels* self.panel = MainPanel(self)  
 self.login = LoginPanel(self)  
 self.register = RegisterPanel(self)  
 self.logic\_q = logic\_q  
 *# self.tree = TreePanel(self, self.logic\_q)* self.SetIcon(wx.Icon(settings.Settings.icon\_path))  
 *# self.panel.Show()* box = wx.BoxSizer(wx.VERTICAL)  
 box.Add(self.panel, 1, wx.EXPAND)  
 box.Add(self.login, 1, wx.EXPAND)  
 box.Add(self.register, 1, wx.EXPAND)  
  
  
 *# arrange the frame* self.SetSizer(box)  
 self.Layout()  
 self.Show()  
  
 self.panel.Show()  
 pub.subscribe(TreeFrame.show\_error, "error")  
  
  
 def change\_panel(self, current: wx.Panel, new\_panel: wx.Panel):  
 *"""  
 switches between panels  
 :param current: current panel working on  
 :param new\_panel: panel to switch to  
 :return: None  
 """* current.Hide()  
 new\_panel.SetSize(current.GetSize())  
 new\_panel.Show()  
  
  
 def go\_to\_tree(self):  
 *"""  
 goes to tree panel  
 :return: None  
 """* self.Hide()  
 TreeFrame(self, self.logic\_q)  
  
  
class MainPanel(wx.Panel):  
 def \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(500, 500))  
 self.frame = parent  
 self.SetBackgroundColour(wx.Colour(30, 30, 30))  
 self.parent = parent  
 sizer = wx.BoxSizer(wx.VERTICAL)  
  
 main\_box = wx.BoxSizer(wx.VERTICAL)  
 logo\_image = wx.Image(os.path.join(settings.Settings.pic\_path, "white\_logo.png"))  
 logo\_image.Rescale(200, 50)  
 logo\_bitmap = wx.Bitmap(logo\_image)  
 logo = wx.StaticBitmap(self, bitmap=logo\_bitmap)  
  
 button\_box = wx.BoxSizer(wx.HORIZONTAL)  
 main\_box.AddSpacer(15)  
 main\_box.Add(logo, 0, wx.CENTER | wx.ALL, 5)  
 main\_box.AddSpacer(20)  
 loginBtn = wx.Button(self, wx.ID\_ANY, label="Login", size=(200, 80))  
 registerBtn = wx.Button(self, wx.ID\_ANY, label="Register", size=(200, 80))  
 loginBtn.Bind(wx.EVT\_BUTTON, self.handle\_login)  
 registerBtn.Bind(wx.EVT\_BUTTON, self.handle\_register)  
 button\_box.Add(loginBtn, 0, wx.ALL, 5)  
 button\_box.Add(registerBtn, 0, wx.ALL, 5)  
 main\_box.Add(button\_box, wx.CENTER | wx.TOP, 5)  
  
 *# add all elements to sizer* sizer.Add(main\_box, wx.CENTER | wx.ALL, 5)  
  
 *# arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
  
  
 def handle\_login(self, evt):  
 *"""  
 handles when client presses logs in  
 :param evt: event  
 :return: None  
 """* self.frame.change\_panel(self, self.frame.login)  
  
 def handle\_register(self, evt):  
 *"""  
 handles when client presses register  
 :param evt: event  
 :return: None  
 """* self.frame.change\_panel(self, self.frame.register)  
  
  
  
  
class RegisterPanel(wx.Panel):  
 def \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(500, 500))  
 self.username = None  
 self.password = None  
 self.parent = parent  
 sizer = wx.BoxSizer(wx.VERTICAL)  
 bg\_colour = wx.Colour(30, 30, 30)  
 *# Define the title label* title\_font = wx.Font(24, wx.FONTFAMILY\_MODERN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL)  
 title\_text = wx.StaticText(self, label="Register")  
 title\_text.SetFont(title\_font)  
 title\_text.SetForegroundColour(wx.Colour(255, 255, 255)) *# Custom text color* nameBox = wx.BoxSizer(wx.HORIZONTAL)  
 self.SetBackgroundColour(wx.Colour(bg\_colour))  
 self.SetForegroundColour(wx.Colour(150, 150, 150))  
 nameText = wx.StaticText(self, 1, label="Username: ")  
  
 self.nameField = wx.TextCtrl(self, -1, name="username", size=(150, -1))  
 self.nameField.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* self.nameField.SetForegroundColour(wx.Colour(255, 255, 255))  
 nameBox.Add(nameText, 0, wx.ALL, 5)  
 nameBox.Add(self.nameField, 0, wx.ALL, 5)  
  
 passBox = wx.BoxSizer(wx.HORIZONTAL)  
 passText = wx.StaticText(self, 1, label="Password: ")  
 *# passText.SetFont()* self.passField = wx.TextCtrl(self, -1, name="password", style=wx.TE\_PASSWORD, size=(150, -1))  
 self.passField.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* self.passField.SetForegroundColour(wx.Colour(255, 255, 255))  
  
 passBox.Add(passText, 0, wx.ALL, 5)  
 passBox.Add(self.passField, 0, wx.ALL, 5)  
  
 btnBox = wx.BoxSizer(wx.HORIZONTAL)  
  
 login\_button = wx.Button(self, label="Register")  
 login\_button.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* login\_button.SetForegroundColour(wx.Colour(255, 255, 255)) *# Custom text color* login\_button.SetFont(wx.Font(12, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD))  
 login\_button.Bind(wx.EVT\_BUTTON, self.handle\_reg)  
  
 back\_btn = wx.Button(self, wx.ID\_ANY, label="Back")  
 back\_btn.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* back\_btn.SetForegroundColour(wx.Colour(255, 255, 255)) *# Custom text color* back\_btn.SetFont(wx.Font(12, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD))  
 back\_btn.Bind(wx.EVT\_BUTTON, self.handle\_back)  
 btnBox.Add(back\_btn, 1, wx.ALL, 5)  
 btnBox.Add(login\_button, 0, wx.ALL, 5)  
  
 *# add all elements to sizer* sizer.AddStretchSpacer(20)  
 sizer.Add(title\_text, 0, wx.CENTER | wx.BOTTOM, border=0)  
  
 sizer.AddSpacer(10)  
 sizer.Add(nameBox, 0, wx.CENTER | wx.ALL, 5)  
 sizer.Add(passBox, -1, wx.CENTER | wx.ALL, 5)  
 sizer.AddSpacer(10)  
 sizer.Add(btnBox, wx.CENTER | wx.ALL, 5)  
 *# arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
  
 pub.subscribe(self.parent.go\_to\_tree, "login")  
 pub.subscribe(self.do\_login, 'register\_ok')  
  
 def handle\_reg(self, event):  
 *"""  
 sends to logic username and password to register  
 :param event: event got  
 :return: None  
 """* self.username = self.nameField.GetValue()  
 self.password = self.passField.GetValue()  
 if not self.username or not self.password:  
 TreeFrame.show\_error('must enter username and password')  
 else:  
 self.parent.logic\_q.put(('register', f'{self.username},{self.password}'))  
  
 def handle\_back(self, evt):  
 *"""  
 gets client back to main panel  
 :param evt: event got  
 :return: None  
 """* self.parent.change\_panel(self, self.parent.panel)  
  
 def do\_login(self):  
 *"""  
 calls login  
 :return: none  
 """* if self.username and self.password:  
 self.parent.logic\_q.put(('login', f'{self.username},{self.password}'))  
  
class LoginPanel(wx.Panel):  
 def \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(500, 500))  
 self.parent = parent  
 sizer = wx.BoxSizer(wx.VERTICAL)  
 bg\_colour = wx.Colour(30, 30, 30)  
 *# Define the title label* title\_font = wx.Font(24, wx.FONTFAMILY\_MODERN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL)  
 title\_text = wx.StaticText(self, label="Login")  
 title\_text.SetFont(title\_font)  
 title\_text.SetForegroundColour(wx.Colour(255, 255, 255)) *# Custom text color* nameBox = wx.BoxSizer(wx.HORIZONTAL)  
 self.SetBackgroundColour(wx.Colour(bg\_colour))  
 self.SetForegroundColour(wx.Colour(150, 150, 150))  
 nameText = wx.StaticText(self, 1, label="Username: ")  
  
 self.nameField = wx.TextCtrl(self, -1, name="username", size=(150, -1))  
 self.nameField.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* self.nameField.SetForegroundColour(wx.Colour(255, 255, 255))  
 nameBox.Add(nameText, 0, wx.ALL, 5)  
 nameBox.Add(self.nameField, 0, wx.ALL, 5)  
  
 passBox = wx.BoxSizer(wx.HORIZONTAL)  
 passText = wx.StaticText(self, 1, label="Password: ")  
 *# passText.SetFont()* self.passField = wx.TextCtrl(self, -1, name="password", style=wx.TE\_PASSWORD, size=(150, -1))  
 self.passField.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* self.passField.SetForegroundColour(wx.Colour(255, 255, 255))  
  
 passBox.Add(passText, 0, wx.ALL, 5)  
 passBox.Add(self.passField, 0, wx.ALL, 5)  
  
 btnBox = wx.BoxSizer(wx.HORIZONTAL)  
  
 login\_button = wx.Button(self, label="Login")  
 login\_button.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* login\_button.SetForegroundColour(wx.Colour(255, 255, 255)) *# Custom text color* login\_button.SetFont(wx.Font(12, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD))  
 login\_button.Bind(wx.EVT\_BUTTON, self.handle\_login)  
  
 back\_btn = wx.Button(self, wx.ID\_ANY, label="Back")  
 back\_btn.SetBackgroundColour(wx.Colour(50, 50, 50)) *# Custom background color* back\_btn.SetForegroundColour(wx.Colour(255, 255, 255)) *# Custom text color* back\_btn.SetFont(wx.Font(12, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD))  
 back\_btn.Bind(wx.EVT\_BUTTON, self.handle\_back)  
 btnBox.Add(back\_btn, 1, wx.ALL, 5)  
 btnBox.Add(login\_button, 0, wx.ALL, 5)  
  
 *# add all elements to sizer* sizer.AddStretchSpacer(20)  
 sizer.Add(title\_text, 0, wx.CENTER | wx.BOTTOM, border=0)  
  
 sizer.AddSpacer(10)  
 sizer.Add(nameBox, 0, wx.CENTER | wx.ALL, 5)  
 sizer.Add(passBox, -1, wx.CENTER | wx.ALL, 5)  
 sizer.AddSpacer(10)  
 sizer.Add(btnBox, wx.CENTER | wx.ALL, 5)  
 *# arrange the screen* self.SetSizer(sizer)  
 self.Layout()  
 self.Hide()  
  
 pub.subscribe(self.parent.go\_to\_tree, "login")  
  
 def handle\_login(self, event):  
 *"""  
 sends to logic login with username and password  
 :param event: event got  
 :return: None  
 """* username = self.nameField.GetValue()  
 password = self.passField.GetValue()  
 if not username or not password:  
 TreeFrame.show\_error('must enter name and password')  
 else:  
 self.parent.logic\_q.put(('login', f'{username},{password}'))  
  
  
 def handle\_back(self, evt):  
 *"""  
 gets back to main panel  
 :param evt: event got  
 :return: None  
 """* self.parent.change\_panel(self, self.parent.panel)  
  
 def go\_to\_tree(self):  
 *"""  
 goes to tree panel  
 :return: None  
 """* self.parent.Hide()  
 TreeFrame(self, self.parent.logic\_q)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 graphic\_q = Queue()  
 a = wx.App(False)  
 f = MyFrame(graphic\_q)  
 f.Show()  
 a.MainLoop()

**tree\_graphic.py:**

import wx  
import os  
from pubsub import pub  
from Reflection import settings  
from queue import Queue  
from Reflection.local\_handler.file\_handler import FileHandler  
from Reflection.settings import Settings  
  
class CreateFileDialog(wx.Dialog):  
 def \_\_init\_\_(self, parent, title: str, is\_folder):  
  
 super(CreateFileDialog, self).\_\_init\_\_(parent, title=title, size=(300, 150))  
 self.is\_folder = is\_folder  
  
 self.file\_name = ''  
 self.panel = wx.Panel(self)  
 vbox = wx.BoxSizer(wx.VERTICAL)  
  
  
 text\_box = wx.BoxSizer(wx.HORIZONTAL)  
 lbl = wx.StaticText(self.panel, label="Enter Name:")  
 text\_box.Add(lbl, flag=wx.RIGHT, border=8)  
 self.text\_ctrl = wx.TextCtrl(self.panel)  
 text\_box.Add(self.text\_ctrl, proportion=1)  
 vbox.Add(text\_box, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)  
  
 btn\_box = wx.BoxSizer(wx.HORIZONTAL)  
 btn\_ok = wx.Button(self.panel, label='OK')  
 btn\_ok.Bind(wx.EVT\_BUTTON, self.on\_ok)  
 btn\_box.Add(btn\_ok)  
 btn\_cancel = wx.Button(self.panel, label='Cancel')  
 btn\_cancel.Bind(wx.EVT\_BUTTON, self.on\_cancel)  
 btn\_box.Add(btn\_cancel, flag=wx.LEFT, border=5)  
 vbox.Add(btn\_box, flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP, border=10)  
  
 self.panel.SetSizer(vbox)  
  
 def on\_ok(self, event):  
 *"""  
 when pressed on ok gets input and tells if valid or not  
 :param event: event happened  
 """* self.file\_name = self.text\_ctrl.GetValue()  
 self.EndModal(wx.ID\_OK)  
  
  
 def on\_cancel(self, event):  
 *"""  
 closes file dialog  
 :param event: event happened  
 """* self.EndModal(wx.ID\_CANCEL)  
  
  
  
class TreeFrame(wx.Frame):  
 open\_folder\_name = 'open\_folder.png'  
 close\_folder\_name = 'fld.png'  
 file\_name = 'txt.png'  
 image\_types = ["apng", "avif", "gif", "jpg", "jpeg", "jfif", "pjpeg", "pjp", "png", "svg", "webp", "bmp", "ico",  
 "cur",  
 "tif", "tiff"]  
 def \_\_init\_\_(self, parent, command\_q: Queue):  
 *# wx.Panel.\_\_init\_\_(self, parent, -1)* wx.Frame.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(500, 500))  
 self.cwd = settings.Settings.pic\_path  
 self.tree = wx.TreeCtrl(self, style=wx.TR\_HIDE\_ROOT)  
 self.file\_commands = ('open', 'delete', 'rename', 'download', 'copy')  
 self.folder\_commands = ('create file', 'create folder', 'delete', 'rename', 'upload file', 'paste')  
 self.root = self.tree.AddRoot("root")  
 self.command\_q = command\_q  
 self.folders = []  
 self.image\_size = 16  
 self.item\_image\_path = {}  
 self.image\_list = wx.ImageList(self.image\_size, self.image\_size)  
 self.tree.AssignImageList(self.image\_list)  
 self.path\_item = {}  
 self.forbidden = ('\*', ',', '\\', '/', '[', ']', '{', '}', '?', '<', '>', ' ', ':', '|', '(', ')', '-')  
 self.on\_clipboard\_path = None  
 self.loading\_cursor = wx.Cursor(wx.CURSOR\_WAIT)  
  
 self.my\_ip = Settings.get\_ip()  
  
 self.font = wx.Font(round(self.image\_size \* 0.7), wx.FONTFAMILY\_MODERN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL)  
 self.tree.SetFont(self.font)  
 self.tree.SetForegroundColour(wx.Colour(255, 255, 255))  
  
 self.tree.Bind(wx.EVT\_TREE\_ITEM\_ACTIVATED, self.on\_double\_clicked)  
 self.tree.Bind(wx.EVT\_TREE\_ITEM\_EXPANDED, self.on\_expanded)  
 self.tree.Bind(wx.EVT\_TREE\_ITEM\_COLLAPSED, self.on\_collapsed)  
 self.tree.Bind(wx.EVT\_TREE\_ITEM\_RIGHT\_CLICK, self.on\_right\_click)  
 self.tree.Bind(wx.EVT\_TREE\_BEGIN\_DRAG, self.on\_drag)  
 self.tree.Bind(wx.EVT\_TREE\_END\_DRAG, self.on\_drop)  
 self.tree.Bind(wx.EVT\_CHAR\_HOOK, self.copy\_paste)  
 self.tree.Bind(wx.EVT\_MOUSEWHEEL, self.change\_size)  
  
 self.SetIcon(wx.Icon(settings.Settings.icon\_path))  
 pub.subscribe(self.convert\_to\_tree, "update\_tree")  
 pub.subscribe(self.add\_object, "create")  
 pub.subscribe(self.delete\_object, "delete")  
 pub.subscribe(self.show\_error, "error")  
 pub.subscribe(self.rename\_object, "rename")  
 pub.subscribe(self.refresh\_cursor, "cursor")  
  
 self.tree.SetBackgroundColour(wx.Colour(30, 30, 30))  
 self.Show()  
  
  
  
 def refresh\_cursor(self):  
 *"""  
 refreshes cursor after finishing action  
 """* self.SetCursor(wx.Cursor())  
  
 def send\_to\_logic(self, command: str, data: str):  
 *"""  
 sends to logic data  
 :param command: the command the logic should do with the data  
 :param data: the data delivered with the command  
 :return None  
 """* my\_cursor = wx.Cursor(wx.CURSOR\_WAIT)  
 self.SetCursor(my\_cursor)  
 self.command\_q.put((command, data))  
  
  
 def change\_size(self, evt: wx.MouseEvent):  
 *"""  
 changes size of font and pictures  
 :param evt: mouse scroll up or down  
 :return: None  
 """* if not evt.CmdDown():  
 evt.Skip()  
 return  
  
 *# bounds* if self.image\_size < 10 or self.image\_size > 30:  
 if self.image\_size < 10:  
 self.image\_size = 10  
 else:  
 self.image\_size = 30  
  
 if evt.GetWheelRotation() > 0:  
 self.image\_size += 1  
 else:  
 self.image\_size -= 1  
  
 *# changing font* self.font.SetPointSize(self.image\_size)  
 self.tree.SetFont(self.font)  
  
 self.image\_list = wx.ImageList(self.image\_size, self.image\_size)  
 self.tree.AssignImageList(self.image\_list)  
  
 *# resizing every image* for item in self.item\_image\_path:  
 path = self.item\_image\_path[item]  
 result = wx.Bitmap(path)  
 image = result.ConvertToImage()  
 image = image.Scale(self.image\_size, self.image\_size, wx.IMAGE\_QUALITY\_HIGH)  
 image = self.image\_list.Add(image.ConvertToBitmap())  
 self.tree.SetItemImage(item, image, wx.TreeItemIcon\_Normal)  
  
  
  
 def add\_pic(self, item, name: str, is\_folder: bool):  
 *"""  
 gets item and adds correct image to it  
 :param item: object in tree  
 :param name: name of item  
 :return: None  
 """* if is\_folder:  
 pic\_name = self.close\_folder\_name  
 else:  
 print(name)  
 typ\_index = name.rfind('.')  
 typ = name[typ\_index + 1:]  
 pic\_name = typ  
 if typ in TreeFrame.image\_types:  
 pic\_name = 'pic'  
 pic\_name += '.png'  
 if pic\_name not in os.listdir(settings.Settings.pic\_path):  
 pic\_name = 'txt.png'  
  
 pic\_path = self.cwd + pic\_name  
 self.put\_pic\_for\_item(pic\_path, item)  
  
 def put\_pic\_for\_item(self, pic\_path: str, item: wx.TreeItemId):  
 *"""  
 changes pic for certain item  
 :param pic\_path: path of the pic  
 :param item: the item in tree  
 :return: None  
 """* item\_image = wx.Image(pic\_path, wx.BITMAP\_TYPE\_PNG).Scale(self.image\_size, self.image\_size)  
 self.item\_image\_path[item] = pic\_path  
 item\_image = item\_image.ConvertToBitmap()  
 item\_image = self.image\_list.Add(item\_image)  
 self.tree.SetItemImage(item, item\_image, wx.TreeItemIcon\_Normal)  
  
  
 def on\_expanded(self, evt):  
 *"""  
 gets evt and does stuff for expanded item  
 :param evt: event happened  
 :return: None  
 """* item = evt.GetItem()  
 pic\_path = self.cwd + self.open\_folder\_name  
 self.put\_pic\_for\_item(pic\_path, item)  
  
 def on\_collapsed(self, evt):  
 *"""  
 gets evt and does stuff for collapsed item  
 :param evt: event happened  
 :return: None  
 """* item = evt.GetItem()  
 pic\_path = self.cwd + self.close\_folder\_name  
 self.put\_pic\_for\_item(pic\_path, item)  
  
 def copy\_paste(self, evt: wx.KeyEvent):  
 *"""  
 get key pressed and tells logic if got copy or paste  
 :param evt: key event  
 :return: None  
 """* key\_code = evt.GetKeyCode()  
 ctrl\_down = evt.CmdDown()  
 if ctrl\_down:  
  
 *# copy* if key\_code == 67:  
 on\_item = self.tree.GetFocusedItem()  
 on\_item\_path = self.tree.GetItemData(on\_item)  
 print('copy')  
 if on\_item\_path in self.folders:  
 self.show\_error('cannot copy folder')  
 else:  
 self.on\_clipboard\_path = on\_item\_path  
  
 *# paste* elif key\_code == 86:  
 on\_item = self.tree.GetFocusedItem()  
 on\_item\_path = self.tree.GetItemData(on\_item)  
 print('paste')  
 if self.on\_clipboard\_path:  
 self.send\_to\_logic('paste', f'{self.on\_clipboard\_path},{on\_item\_path}')  
  
 def on\_drop(self, evt):  
 *"""  
 tells graphic when drag stopped and where  
 :param evt: event got  
 :return: None  
 """* self.refresh\_cursor()  
 dropped\_on = self.tree.GetItemData(evt.GetItem())  
 dragged\_data = self.tree.GetItemData(self.drag\_item)  
 self.send\_to\_logic('move', f'{dragged\_data},{dropped\_on}')  
  
 def on\_drag(self, evt):  
 *"""  
 lets drag start visually and keeps item dragged  
 :param evt: event got  
 :return: None  
 """* evt.Allow()  
 self.drag\_item = evt.GetItem()  
 cursor = wx.Cursor(wx.Image(self.item\_image\_path[self.drag\_item]))  
 self.SetCursor(cursor)  
 print(self.drag\_item)  
  
 def valid\_input(self, file\_name: str, is\_folder: bool):  
 *"""  
 check if input is up to standard  
 :return: if input is valid or not  
 """* valid = True  
 *# temp* forbidden = list(self.forbidden)  
  
 if is\_folder:  
 forbidden.append('.')  
 else:  
 valid = file\_name.count('.') == 1  
 for bad in forbidden:  
 if bad in file\_name or not valid:  
 valid = False  
 break  
  
 if len(file\_name) > 30:  
 valid = False  
 return valid  
  
  
 def create\_file\_dialog(self, command: str, path: str):  
 *"""  
 creates file dialog and sends answer to logic  
 :param command: command  
 :param path:  
 :return:  
 """* dialog\_title = command + ' '  
 if command.startswith('create'):  
 dialog\_title += f'in {os.path.basename(path)}:'  
 is\_folder = command.endswith('folder')  
 elif command == 'rename':  
 dialog\_title += f'{os.path.basename(path)} to:'  
 is\_folder = path in self.folders  
  
 else:  
 *# temp* is\_folder = False  
 while True:  
 dialog = CreateFileDialog(self, dialog\_title.capitalize(), is\_folder)  
 if dialog.ShowModal() == wx.ID\_OK:  
 file\_name = dialog.file\_name  
 full\_path = os.path.join(path, file\_name)  
 print('full\_path:', full\_path)  
 if not self.valid\_input(file\_name, is\_folder):  
 self.show\_error(f'name "{file\_name}" is not valid')  
 elif full\_path in self.path\_item:  
 self.show\_error(f'"{file\_name}" already exists in this directory')  
 else:  
 self.send\_to\_logic(command, full\_path)  
 break  
 else:  
 break  
 dialog.Destroy()  
  
 def rename\_object(self, old\_path: str, new\_name: str):  
 *"""  
 renames an object in the tree  
 :param old\_path: the old path of the object  
 :param new\_name: the new name of the object  
 :return if success in renaming object  
 """* worked = old\_path in self.path\_item  
 if worked:  
 item = self.path\_item[old\_path]  
 self.tree.SetItemText(item, new\_name)  
  
 folder\_path, \_ = FileHandler.split\_path\_last\_part(old\_path)  
 new\_path = os.path.join(folder\_path, new\_name)  
  
 to\_add = {}  
 to\_remove = []  
 for path in self.path\_item.keys():  
 print('path: ', path)  
 print('old\_path:', old\_path)  
 if path.startswith(old\_path):  
 print('got in')  
 item = self.path\_item[path]  
 to\_remove.append(path)  
 replaced\_path = path.replace(old\_path, new\_path, 1)  
 print('replaced\_path:', replaced\_path)  
 to\_add[replaced\_path] = item  
 self.tree.SetItemData(item, replaced\_path)  
  
 print()  
 for path in to\_add:  
 self.path\_item[path] = to\_add[path]  
  
 for path in to\_remove:  
 del self.path\_item[path]  
  
 print('path\_item\_before:', self.path\_item.keys())  
 print('path\_item\_after:', self.path\_item.keys())  
 *# is folder* if '.' not in new\_name:  
 self.folders.remove(old\_path)  
 self.folders.append(new\_path)  
  
 return worked  
  
 def delete\_object(self, path: str):  
 *"""  
 gets path of object and deletes it from needed places  
 :param path: path of object  
 """* if path in self.path\_item:  
 self.tree.Delete(self.path\_item[path])  
 del self.path\_item[path]  
  
 def add\_object(self, path: str, name: str, typ: str):  
 *"""  
 adds object to tree  
 :param path: path of file (directory above it)  
 :param name: name of file  
 :param typ: type of file\folder  
 """* print(f'path: {path}')  
 print(f'path\_item: {self.path\_item}')  
 if path not in self.path\_item:  
 return  
  
 dir\_on = self.path\_item[path]  
 full\_path = os.path.join(path, name)  
 if typ == 'fld':  
  
 new\_item = self.tree.AppendItem(dir\_on, f'{name}', data=full\_path)  
 self.path\_item[full\_path] = new\_item  
 self.folders.append(full\_path)  
 self.add\_pic(new\_item, name, True)  
 else:  
 full\_path = f'{full\_path}.{typ}'  
 new\_item = self.tree.AppendItem(dir\_on, f'{name}.{typ}', data=full\_path)  
 self.path\_item[full\_path] = new\_item  
 self.add\_pic(new\_item, f'{name}.{typ}', False)  
  
  
 def on\_right\_click(self, evt):  
 *"""  
 gets evt and does stuff for right click  
 :param evt: event happened  
 :return: None  
 """* item = evt.GetItem()  
 self.tree.SetFocusedItem(item)  
 item\_path = self.tree.GetItemData(item)  
 if item\_path in self.folders:  
 commands = self.folder\_commands  
 else:  
 commands = self.file\_commands  
  
 self.popupmenu = wx.Menu()  
 item\_name = os.path.basename(item\_path)  
 self.popupmenu.SetTitle(f'{item\_name}:')  
 for command in commands:  
 self.popupmenu.Append(-1, command)  
 self.Bind(wx.EVT\_MENU, lambda event, item\_pressed=item: self.command\_selected(event, item\_pressed))  
  
 self.PopupMenu(self.popupmenu)  
  
 def command\_selected(self, evt: wx.CommandEvent, item):  
 *"""  
 gets event and item and informs logic which command was chosen and on what  
 :param evt: event happened  
 :param item: item pressed on  
 :return: None  
 """* id\_selected = evt.GetId()  
 obj = evt.GetEventObject()  
 text = obj.GetLabel(id\_selected)  
 path = self.tree.GetItemData(item)  
 if text.startswith('create') or text == 'rename':  
 self.create\_file\_dialog(text, path)  
 elif text == 'upload file' or text == 'download':  
 file\_dialog = wx.FileDialog(None, "Choose a file", style=wx.FD\_OPEN)  
 file\_explorer\_path\_chose = None  
 if file\_dialog.ShowModal() == wx.ID\_OK:  
 file\_explorer\_path\_chose = file\_dialog.GetPath()  
 print("Selected file:", file\_explorer\_path\_chose)  
  
 file\_dialog.Destroy()  
 if file\_explorer\_path\_chose:  
 self.send\_to\_logic(text, f'{path},{file\_explorer\_path\_chose}')  
  
 elif text == 'paste':  
 on\_item = self.tree.GetFocusedItem()  
 on\_item\_path = self.tree.GetItemData(on\_item)  
 if self.on\_clipboard\_path:  
 self.send\_to\_logic(text, f'{self.on\_clipboard\_path},{on\_item\_path}')  
  
 elif text == 'copy':  
 on\_item = self.tree.GetFocusedItem()  
 on\_item\_path = self.tree.GetItemData(on\_item)  
 self.on\_clipboard\_path = on\_item\_path  
  
 else:  
 self.send\_to\_logic(text, path)  
  
 def convert\_to\_tree(self, dic: dict, father=None):  
 *"""  
 gets dictionary and converts it to tree  
 :param dic: dictionary  
 :param father: a parent  
 :return: None  
 """* if not father:  
 print('dictionary working with:', dic)  
 father\_path = next(iter(dic.keys()))  
 father = self.root  
 else:  
 father\_path = self.tree.GetItemData(father)  
  
 folder = True  
 if father\_path not in dic:  
 return  
  
 *# runs on every object in this folder adds it to tree and if is folder then* for element in dic[father\_path]:  
 if element == ',':  
 folder = False  
 continue  
  
 path = os.path.join(father\_path, element)  
 if element == self.my\_ip:  
 element = 'my\_pc'  
  
 new\_item = self.tree.AppendItem(father, element, data=path)  
  
 print('first:', self.tree.GetItemData(self.tree.GetFirstVisibleItem()))  
 self.path\_item[path] = new\_item  
 self.add\_pic(new\_item, element, folder)  
 if folder:  
 self.folders.append(path)  
 self.convert\_to\_tree(dic, new\_item)  
  
  
 def on\_double\_clicked(self, evt):  
 *"""  
 gets evt and does stuff for double clicked  
 :param evt: event happened  
 :return: None  
 """* item = evt.GetItem()  
 path = self.tree.GetItemData(item)  
 if path in self.folders:  
 if self.tree.IsExpanded(item):  
 self.tree.Collapse(item)  
 else:  
 self.tree.Expand(item)  
  
 *# open file* else:  
 self.send\_to\_logic('open', path)  
  
  
 @staticmethod  
 def show\_error(error: str):  
 *"""  
 gets error and tells it to user  
 :param error: the error explanation  
 """* wx.MessageBox(error, "Error", wx.OK | wx.ICON\_ERROR)  
  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = wx.App(False)  
 first = TreeFrame(None, Queue())  
 app.MainLoop()

**main\_server.py:**

import queue  
from Reflection.settings import Settings  
from Reflection.comms import server\_comm  
from Reflection.database import db  
from Reflection.protocols import server\_protocol as protocol  
from Reflection.encryption import symmetrical\_encryption  
from Reflection.local\_handler.file\_handler import FileHandler  
import os  
  
  
  
def handle\_disconnect(client\_ip: str, called\_by\_server\_comm: bool):  
 *"""  
 disconnect client by ip and removes from dictionaries  
 :param client\_ip: ip address  
 :param called\_by\_server\_comm: if was called by the server comm  
 :return: None  
 """  
  
 # calls to delete ip to all users related to ip* for ip in user\_comps:  
 if client\_ip in user\_comps[ip]:  
 username = get\_key\_by\_value(username\_ip, ip)  
 server\_comm.send(ip, protocol.pack\_status\_delete(True, f'{Settings.root}{username}\\{client\_ip[0]}'))  
  
 if client\_ip in ip\_mac:  
 del ip\_mac[client\_ip]  
  
 if client\_ip in user\_comps:  
 del user\_comps[client\_ip]  
  
 for ip in user\_comps:  
 if client\_ip in user\_comps[ip]:  
 user\_comps[ip].remove(client\_ip)  
  
 if client\_ip in username\_ip.values():  
 for key in username\_ip.keys():  
 if username\_ip[key] == client\_ip:  
 del username\_ip[key]  
 break  
  
 if not called\_by\_server\_comm:  
 server\_comm.disconnect\_client(client\_ip, True)  
  
def handle\_register(ip: str, vars: list):  
 *"""  
 gets ip and vars and adds user to db and returns response by protocol  
 :param ip: ip  
 :param vars: user, password  
 :return:  
 """* if len(vars) != 2:  
 handle\_disconnect(ip, False)  
 return  
 user, password = vars  
 to\_send = protocol.pack\_status\_register(db.add\_user(user, password))  
 server\_comm.send(ip, to\_send)  
  
  
def handle\_sign\_in(user\_ip: str, vars: list):  
 *"""  
 signs in the user and returns response accordingly  
 :param user\_ip: user's ip  
 :param vars: username, password, mac address  
 :return: None  
 """* print('got to sign in')  
 if len(vars) != 3:  
 handle\_disconnect(user\_ip, False)  
 return  
  
 user, password, user\_mac = vars  
 db\_pass = db.get\_password(user)  
 hashed\_pass = symmetrical\_encryption.SymmetricalEncryption.hash(password)  
 if db\_pass == hashed\_pass:  
  
 db.add\_user\_mac(user, user\_mac)  
 server\_comm.send(user\_ip, protocol.pack\_status\_login(True))  
 macs\_worked\_on = db.get\_macs(user)  
 print('macs\_worked\_on:', macs\_worked\_on)  
 user\_comps[user\_ip] = [(user\_ip[0], 'G')]  
 username\_ip[user] = user\_ip  
  
 *# adding to user comps* for mac in macs\_worked\_on:  
 for ip in ip\_mac:  
 if ip\_mac[ip] == mac:  
 user\_comps[user\_ip].append(ip)  
 server\_comm.send(ip, protocol.pack\_ask\_file\_Tree(f'{user}'))  
  
 print('user\_comps:', user\_comps)  
 *# asking file tree from own user mac* if user\_mac not in macs\_worked\_on:  
 server\_comm.send((user\_ip[0], 'G'), protocol.pack\_ask\_file\_Tree(f'{user}'))  
  
 else:  
 server\_comm.send(user\_ip, protocol.pack\_status\_login(False))  
  
  
  
def handle\_got\_file\_tree(got\_ip, vars: list):  
 *"""  
  
 :param got\_ip: the ip got file tree from  
 :param vars: file tree got from got\_ip  
 :return: None  
 """* file\_tree = vars[0]  
 got\_ok = True  
 ip\_to\_send = None  
 folders\_with\_ip = {}  
 if len(vars) == 1:  
 user\_to\_send = FileHandler.get\_user(list(file\_tree.keys())[0])  
 print('user\_to\_send:', user\_to\_send)  
 if user\_to\_send in username\_ip:  
 ip\_to\_send = username\_ip[user\_to\_send]  
 print('to\_send:', ip\_to\_send)  
  
 else:  
 got\_ok = False  
 ip\_to\_send = None  
  
 if got\_ok and ip\_to\_send in user\_comps and got\_ip in user\_comps[ip\_to\_send]:  
  
 folders\_with\_ip = {}  
  
 *# inserting ip to each path in dictionary* for folder\_path in file\_tree.keys():  
 user\_plus\_ip = os.path.join(user\_to\_send, got\_ip[0])  
 if user\_to\_send in folder\_path:  
 new\_folder\_path = folder\_path.replace(user\_to\_send, user\_plus\_ip, 1)  
 folders\_with\_ip[new\_folder\_path] = file\_tree[folder\_path]  
 else:  
 got\_ok = False  
 break  
  
 else:  
 got\_ok = False  
 else:  
 got\_ok = False  
  
 if got\_ok and ip\_to\_send:  
 *# sending file tree to user* server\_comm.send(ip\_to\_send, protocol.pack\_send\_file\_tree(folders\_with\_ip))  
  
 else:  
 print('error in getting file tree')  
 handle\_disconnect(got\_ip, False)  
  
  
def handle\_got\_mac(client\_ip: str, vars: list):  
 *"""  
 get mac from ip adds it to ip\_mac and ip to user\_comps and sends status back  
 :param client\_ip: client's ip  
 :param vars: mac address  
 :return: None  
 """* if len(vars) != 1:  
 handle\_disconnect(client\_ip, False)  
 return  
  
 mac = vars[0]  
 users\_for\_mac = db.get\_users(mac)  
 has\_users = len(users\_for\_mac) > 0  
 print(f'has users for {mac}:', has\_users)  
 ip\_mac[client\_ip] = mac  
 if has\_users:  
 print('nice')  
 ip\_for\_mac = [username\_ip[user] for user in users\_for\_mac if user in username\_ip]  
 for ip in ip\_for\_mac:  
 if ip in user\_comps:  
 user\_comps[ip].append(client\_ip)  
  
 *# get user by ip* user = get\_key\_by\_value(username\_ip, ip)  
 server\_comm.send(client\_ip, protocol.pack\_ask\_file\_Tree(f'{user}'))  
  
 server\_comm.send(client\_ip, protocol.pack\_status\_mac(has\_users))  
  
  
def got\_create(got\_ip: str, vars: str):  
 *"""  
 sending appropriate client to create object  
 :param got\_ip: ip sent from  
 :param vars: location to create and typ of creation  
 :return: None  
 """* if len(vars) != 2:  
  
 handle\_disconnect(got\_ip, False)  
 return  
  
 location, typ = vars  
 user\_got = get\_key\_by\_value(username\_ip, got\_ip)  
 ip\_to\_send = FileHandler.extract\_ip(user\_got, location)  
 ip\_to\_send = (ip\_to\_send, 'G')  
 location = FileHandler.remove\_ip(user\_got, location)  
 server\_comm.send(ip\_to\_send, protocol.pack\_do\_create(location, typ))  
  
  
def got\_rename(got\_ip: str, vars: str):  
 *"""  
 sending appropriate client to rename object  
 :param got\_ip: ip sent from  
 :param vars: location of object to rename and name to rename it to  
 :return: None  
 """* if len(vars) != 2:  
  
 handle\_disconnect(got\_ip, False)  
 return  
  
 location, new\_name = vars  
 user\_got = get\_key\_by\_value(username\_ip, got\_ip)  
 ip\_to\_send = FileHandler.extract\_ip(user\_got, location)  
 ip\_to\_send = (ip\_to\_send, 'G')  
 location = FileHandler.remove\_ip(user\_got, location)  
 server\_comm.send(ip\_to\_send, protocol.pack\_do\_rename(location, new\_name))  
  
def got\_move(got\_ip: str, vars: str):  
 *"""  
 sending appropriate client to move object  
 :param got\_ip: ip sent from  
 :param vars: location of object to move and place to move to  
 :return: None  
 """* if len(vars) != 2:  
  
 handle\_disconnect(got\_ip, False)  
 return  
  
 move\_from, move\_to = vars  
 user\_got = get\_key\_by\_value(username\_ip, got\_ip)  
 ip\_from = FileHandler.extract\_ip(user\_got, move\_from)  
 ip\_to\_send = (ip\_from, 'G')  
 server\_comm.send(ip\_to\_send, protocol.pack\_do\_move(move\_from, move\_to))  
  
def got\_clone(got\_ip: str, vars: str):  
 *"""  
 sending appropriate client to clone object  
 :param got\_ip: ip sent from  
 :param vars: location of object to clone and place to clone to  
 :return: None  
 """* if len(vars) != 2:  
 handle\_disconnect(got\_ip, False)  
 return  
  
 copy\_from, copy\_to = vars  
 user\_got = get\_key\_by\_value(username\_ip, got\_ip)  
 ip\_from = FileHandler.extract\_ip(user\_got, copy\_from)  
 ip\_to\_send = (ip\_from, 'G')  
 server\_comm.send(ip\_to\_send, protocol.pack\_do\_clone(copy\_from, copy\_to))  
  
  
def handle\_status\_rename(got\_ip: str, vars: str):  
 *"""  
 sending user if renaming was success  
 :param got\_ip: ip got from  
 :param vars: status, location, new name  
 :return: None  
 """* if len(vars) != 3:  
 handle\_disconnect(got\_ip, False)  
 return  
  
 status, location, new\_name = vars  
 print('status:', status)  
 username = location.replace(FileHandler.root, '')  
 username = username[:username.index('\\')]  
  
 ip\_to\_send = username\_ip[username]  
 location = FileHandler.insert\_ip(location, username, got\_ip[0])  
 status = status == 'ok'  
 server\_comm.send(ip\_to\_send, protocol.pack\_status\_rename(status, location, new\_name))  
  
def handle\_status\_move(got\_ip: str, vars: str):  
 *"""  
 sending user if moving was success  
 :param got\_ip: ip got from  
 :param vars: status, old\_path, new\_path  
 :return: None  
 """* if len(vars) != 3:  
 handle\_disconnect(got\_ip, False)  
 return  
  
 status, move\_from, move\_to = vars  
  
 username = FileHandler.get\_user(move\_from)  
  
 ip\_to\_send = username\_ip[username]  
 status = status == 'ok'  
 server\_comm.send(ip\_to\_send, protocol.pack\_status\_move(status, move\_from, move\_to))  
  
  
def handle\_status\_clone(got\_ip: str, vars: str):  
 *"""  
 sending user if cloning was success  
 :param got\_ip: ip got from  
 :param vars: status, file\_cloned, new path of file cloned  
 :return: None  
 """* if len(vars) != 3:  
 handle\_disconnect(got\_ip, False)  
 return  
  
 status, copy\_from, copy\_to = vars  
  
 username = FileHandler.get\_user(copy\_from)  
 ip\_to\_send = username\_ip[username]  
 status = status == 'ok'  
 server\_comm.send(ip\_to\_send, protocol.pack\_status\_clone(status, copy\_from, copy\_to))  
  
def handle\_status\_create(got\_ip: str, vars: str):  
 *"""  
 sending user if creation was success  
 :param got\_ip: ip got from  
 :param vars: status, location, typ  
 :return: None  
 """* if len(vars) != 3:  
 handle\_disconnect(got\_ip, False)  
 return  
  
 status, location, typ = vars  
 print('location:', location)  
 username = location.replace(FileHandler.root, '')  
 username = username[:username.index('\\')]  
 print('user:', username)  
  
 ip\_to\_send = username\_ip[username]  
 location = FileHandler.insert\_ip(location, username, got\_ip[0])  
 status = status == 'ok'  
 server\_comm.send(ip\_to\_send, protocol.pack\_status\_create(status, location, typ))  
  
def handle\_status\_delete(got\_ip: str, vars: str):  
 *"""  
 sending user if delete was success  
 :param got\_ip: ip got from  
 :param vars: status, location  
 :return: None  
 """* if len(vars) != 2:  
 handle\_disconnect(got\_ip, False)  
 return  
  
 status, location = vars  
 username = location.replace(FileHandler.root, '')  
 username = username[:username.index('\\')]  
  
 ip\_to\_send = username\_ip[username]  
 location = FileHandler.insert\_ip(location, username, got\_ip[0])  
 status = status == 'ok'  
 server\_comm.send(ip\_to\_send, protocol.pack\_status\_delete(status, location))  
  
  
def get\_key\_by\_value(dic: dict, value: str):  
 *"""  
 gets value and returns key of value  
 :param value: value in dic  
 :param dic: the dictionary to get the key from  
 :return: key of value  
 """* return next(filter(lambda item: item[1] == value, dic.items()), None)[0]  
  
  
def got\_delete(got\_ip: str, vars: list):  
 *"""  
 sending appropriate client to delete object  
 :param got\_ip: ip sent from  
 :param vars: location to delete  
 :return: None  
 """* if len(vars) != 1:  
  
 handle\_disconnect(got\_ip, False)  
 return  
  
 location = vars[0]  
 user\_got = get\_key\_by\_value(username\_ip, got\_ip)  
 ip\_to\_send = FileHandler.extract\_ip(user\_got, location)  
 ip\_to\_send = (ip\_to\_send, 'G')  
 location = FileHandler.remove\_ip(user\_got, location)  
 server\_comm.send(ip\_to\_send, protocol.pack\_do\_delete(location))  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 rcv\_q = queue.Queue()  
 server\_port = 2000  
 server\_comm = server\_comm.ServerComm(server\_port, rcv\_q, 6)  
 commands = {'01': handle\_register, '03': handle\_sign\_in, '06': got\_create, '08': got\_rename, '10': got\_delete,  
 '12': got\_move, '14': got\_clone, '20': handle\_got\_file\_tree, '22': handle\_status\_rename,  
 '24': handle\_status\_delete, '26': handle\_status\_move, '28': handle\_status\_clone, '29': handle\_got\_mac,  
 '33': handle\_status\_create}  
 ip\_mac = {}  
 user\_comps = {}  
 username\_ip = {}  
 db = db.Db()  
 while True:  
  
 got = rcv\_q.get()  
 ip, data = got  
 print('data:', data)  
 if data == 'close':  
 handle\_disconnect(ip, True)  
 continue  
  
 data = protocol.unpack(data)  
 if not data or not ip:  
 print('got error from server\_comm')  
 continue  
  
 opcode, params = data  
 if opcode in commands:  
 commands[opcode](ip, params)  
 else:  
 print(f'opcode {opcode} is not on list')  
 handle\_disconnect(ip, False)

**main\_general\_client.py:**

import queue  
import threading  
import time  
  
from Reflection.settings import Settings  
from Reflection.comms.client\_comm import ClientComm  
from Reflection.protocols import general\_client\_protocol as client\_protocol  
from Reflection.protocols import server\_protocol  
import os  
from Reflection.local\_handler.file\_handler import FileHandler  
from Reflection.comms.server\_comm import ServerComm  
from queue import Queue  
  
  
def rcv\_comm(comm, q):  
 *"""  
 gets data from server or clients and calls functions accordingly  
 :param comm: client or server comm  
 :param q: msg q  
 """* commands = {'23': handle\_delete, '16': handle\_open\_file, '21': handle\_rename, '25': move\_from\_server,  
 '27': clone\_from\_server, '18': handle\_changed\_file, '31': handle\_asked\_file\_tree,  
 '32': handle\_create, '34': handle\_status\_mac, '36':clone\_from\_client, '37': handle\_status\_clone,  
 '38': move\_from\_client, '39': handle\_status\_move}  
 while True:  
 is\_server = isinstance(comm, ServerComm)  
 if is\_server:  
 ip, data = q.get()  
 if data == 'close':  
 continue  
 data = server\_protocol.unpack(data)  
  
 else:  
 ip = None  
 data = client\_protocol.unpack(q.get())  
 if not data:  
 print('got None from protocol')  
 continue  
  
 print('data got:', data)  
 opcode, params = data  
  
 *# ending thread listening* if opcode == '00':  
 break  
  
 if is\_server:  
 commands[opcode](ip, comm, params)  
 else:  
 commands[opcode](comm, params)  
  
def handle\_changed\_file(got\_ip: str, server: ServerComm, vars: list):  
 *"""  
 sending status if worked or not to client  
 :param server: server comm  
 :param vars: status  
 :return: None  
 """* if len(vars) != 2:  
 print('error while saving file')  
 return  
 status = (vars[0] == 'ok')  
 path = vars[1]  
 server.send(got\_ip, client\_protocol.pack\_status\_change\_file(status, path))  
  
def handle\_open\_file(got\_ip: str, server: ServerComm, vars: list):  
 *"""  
 getting server comm and path of file to open and sends its data  
 :param server: server comm  
 :param vars: path  
 :return: None  
 """* if len(vars) != 1:  
 print('error in opening file')  
 return  
  
 path = vars[0]  
 user = FileHandler.get\_user(path)  
 local\_path = FileHandler.remove\_ip(user, path)  
 if os.path.isfile(local\_path):  
 with open(local\_path, 'rb') as f:  
 file = f.read()  
  
 print('path of file to open:', path)  
 header = client\_protocol.pack\_status\_open\_file(True, path)  
 server.send\_file(got\_ip, header, file)  
  
 else:  
 server.send(got\_ip, client\_protocol.pack\_status\_open\_file(False))  
  
def handle\_create(client\_got: ClientComm, vars: list):  
 *"""  
 creates what server sent if possible and returns status  
 :param client\_got: client comm  
 :param vars: location of creation and it's type  
 :return: None  
 """* location, typ = vars  
 status = FileHandler.create(location, typ)  
 client\_got.send(client\_protocol.pack\_status\_create(status, location, typ))  
  
def handle\_rename(client\_got: ClientComm, vars: list):  
 *"""  
 renames a file\folder  
 :param client\_got: client comm  
 :param vars: location of object to rename and the new name  
 :return: None  
 """* location, new\_name = vars  
 status = FileHandler.rename(location, new\_name)  
 client\_got.send(client\_protocol.pack\_status\_rename(status, location, new\_name))  
  
  
def clone\_from\_server(client\_got: ClientComm, vars: list):  
 *"""  
 clones a file  
 :param client\_got: client comm  
 :param vars: location of object to clone and the new location  
 :return: None  
 """* copy\_from, copy\_to = vars  
 username = FileHandler.get\_user(copy\_from)  
  
 copy\_to\_ip = FileHandler.extract\_ip(username, copy\_to)  
 my\_ip = FileHandler.extract\_ip(username, copy\_from)  
  
 *# locally cloning file* if my\_ip == copy\_to\_ip:  
 local\_copy\_from = FileHandler.remove\_ip(username, copy\_from)  
 local\_copy\_to = FileHandler.remove\_ip(username, copy\_to)  
  
 status = FileHandler.direct\_copy\_file(local\_copy\_from, local\_copy\_to)  
 client\_got.send(client\_protocol.pack\_status\_clone\_to\_server(status, copy\_from, copy\_to))  
  
 *# sending file to other computer* else:  
 rcv\_q = Queue()  
  
 comm = ClientComm(copy\_to\_ip, Settings.pear\_port, rcv\_q, 8, 'G')  
 threading.Thread(target=rcv\_comm, args=(comm, rcv\_q), daemon=True).start()  
 local\_file\_data = FileHandler.remove\_ip(username, copy\_from)  
 with open(local\_file\_data, 'rb') as f:  
 file\_data = f.read()  
  
 header = client\_protocol.pack\_do\_clone(copy\_to, copy\_from)  
 comm.send\_file(header, file\_data)  
  
  
def clone\_from\_client(got\_ip: str, server: ServerComm, vars: list):  
 *"""  
 saves file in right position after cloned and informing client  
 :param got\_ip: the ip got the file from  
 :param server: the comm  
 :param vars:  
 """* print('got\_ip:', got\_ip)  
 if len(vars) != 3:  
 print('amount of vars is not valid')  
 server.disconnect\_client(got\_ip, True)  
 return  
  
 status, copy\_to, copy\_from = vars  
 status = (status == 'ok')  
 server.send(got\_ip, client\_protocol.pack\_status\_clone\_to\_client(status, copy\_to, copy\_from))  
  
  
def handle\_status\_clone(client\_got: ClientComm, vars: list):  
 *"""  
 informing server if cloned file worked  
 :param client\_got: the comm  
 :param vars: status, new\_path, old\_path  
 """* if len(vars) != 3:  
 print('amount of vars is not valid')  
 client\_got.close()  
 return  
  
 status, copied\_to, copied\_from = vars  
 status = (status == 'ok')  
  
 *# informing server and closing connection to client* client.send(client\_protocol.pack\_status\_clone\_to\_server(status, copied\_from, copied\_to))  
 client\_got.close()  
  
def handle\_status\_move(client\_got: ClientComm, vars: list):  
 *"""  
 informing server if moved file worked and finishing moving if needed  
 :param client\_got: the comm  
 :param vars: status, new\_path, old\_path  
 """* if len(vars) != 3:  
 print('amount of vars is not valid')  
 client\_got.close()  
 return  
  
 status, moved\_to, moved\_from = vars  
 status = (status == 'ok')  
  
 *# completing moving* if status:  
 username = FileHandler.get\_user(moved\_from)  
 local\_moved\_from = FileHandler.remove\_ip(username, moved\_from)  
 FileHandler.delete(local\_moved\_from)  
  
 *# informing server and closing connection to client* client.send(client\_protocol.pack\_status\_move\_to\_server(status, moved\_from, moved\_to))  
 client\_got.close()  
  
def move\_from\_client(got\_ip: str, server: ServerComm, vars: list):  
 *"""  
 saves file in right position after moved and informing client  
 :param got\_ip: the ip got the file from  
 :param server: the comm  
 :param vars: status, move\_to, move\_from  
 """* print('got\_ip:', got\_ip)  
 if len(vars) != 3:  
 print('amount of vars is not valid')  
 server.disconnect\_client(got\_ip, True)  
 return  
  
 status, move\_to, move\_from = vars  
 status = (status == 'ok')  
 server.send(got\_ip, client\_protocol.pack\_status\_move\_to\_client(status, move\_to, move\_from))  
  
def move\_from\_server(client\_got: ClientComm, vars: list):  
 *"""  
 moves a file  
 :param client\_got: client comm  
 :param vars: location of object to move and the new location  
 :return: None  
 """* move\_from, move\_to = vars  
 username = FileHandler.get\_user(move\_from)  
  
 move\_to\_ip = FileHandler.extract\_ip(username, move\_to)  
 my\_ip = FileHandler.extract\_ip(username, move\_from)  
  
 *# locally moving file* if my\_ip == move\_to\_ip:  
 local\_move\_from = FileHandler.remove\_ip(username, move\_from)  
 local\_move\_to = FileHandler.remove\_ip(username, move\_to)  
  
 status = FileHandler.move(local\_move\_from, local\_move\_to)  
 client\_got.send(client\_protocol.pack\_status\_move\_to\_server(status, move\_from, move\_to))  
  
 *# sending file to other computer* else:  
 rcv\_q = Queue()  
  
 comm = ClientComm(move\_to\_ip, Settings.pear\_port, rcv\_q, 8, 'G')  
 threading.Thread(target=rcv\_comm, args=(comm, rcv\_q), daemon=True).start()  
 local\_file\_data = FileHandler.remove\_ip(username, move\_from)  
  
 if os.path.isfile(local\_file\_data):  
 with open(local\_file\_data, 'rb') as f:  
 file\_data = f.read()  
  
 header = client\_protocol.pack\_do\_move(move\_to, move\_from)  
 comm.send\_file(header, file\_data)  
 else:  
 print('something went wrong')  
  
  
def handle\_delete(client\_got: ClientComm, vars: list):  
 *"""  
 deletes what server sent if possible and returns status to server  
 :param client\_got: client comm  
 :param vars: location to delete  
 :return: None  
 """* location = vars[0]  
 status = FileHandler.delete(location)  
 client\_got.send(client\_protocol.pack\_status\_delete(status, location))  
  
  
def send\_mac():  
 *"""  
 sends the mac to server  
 """* to\_send = client\_protocol.pack\_mac(Settings.get\_mac\_address())  
 client.send(to\_send)  
  
  
def handle\_status\_mac(client\_got: ClientComm, vars: list):  
 *"""  
 gets status and shows user what happened  
 :param client\_got: the client got  
 :param vars: success or failure  
 :return: None  
 """* status = vars[0]  
 if status == 'ok':  
 print('sent mac successfully')  
 else:  
 print('no computers with users using this pc')  
  
  
def handle\_asked\_file\_tree(client\_got: ClientComm, vars: list):  
 *"""  
 gets folder and sends server file tree of folder  
 :param client\_got: the comm got  
 :param vars: folder name  
 :return: None  
 """* print('got to handle\_asked\_file\_tree')  
 folder = vars[0]  
 folder\_path = f'{FileHandler.root}{folder}'  
 print(folder\_path)  
 time.sleep(1)  
 print('folder\_path:', folder\_path)  
 if os.path.isdir(folder\_path):  
 client\_got.send(client\_protocol.pack\_file\_tree(folder\_path))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 client\_rcv\_q = queue.Queue()  
 server\_rcv\_q = queue.Queue()  
 server\_ip = Settings.server\_ip  
 port = 2000  
 client = ClientComm(server\_ip, port, client\_rcv\_q, 6, 'G')  
 file\_server = ServerComm(Settings.pear\_port, server\_rcv\_q, 8)  
   
 threading.Thread(target=rcv\_comm, args=(client, client\_rcv\_q,), daemon=True).start()  
  
 *# sending main server client's mac* send\_mac()  
  
 *# waiting for pear to pear action* rcv\_comm(file\_server, server\_rcv\_q)

**main\_user\_client.py:**

from queue import Queue  
import threading  
from Reflection.comms.client\_comm import ClientComm  
from Reflection.protocols import user\_client\_protocol as protocol  
from uuid import getnode  
import os  
from Reflection.local\_handler.file\_handler import FileHandler  
from Reflection.local\_handler import process\_handler  
from Reflection.settings import Settings  
import wx  
from pubsub import pub  
from Reflection.graphics.graphics import MyFrame  
import win32file  
from functools import partial  
  
class MainUserClient:  
  
 def \_\_init\_\_(self):  
 self.file\_tree = {}  
 self.server\_rcv\_q = Queue()  
 self.client = ClientComm(Settings.server\_ip, Settings.server\_port, self.server\_rcv\_q, 6, 'U')  
 self.user\_name = ''  
 self.handle\_tree = Queue()  
 self.ip\_comm = {}  
 self.graphic\_q = Queue()  
 self.downloads = {} *# {local path, path on gui}* threading.Thread(target=self.rcv\_comm, args=(self.client,), daemon=True).start()  
 threading.Thread(target=self.rcv\_graphic, args=(self.graphic\_q,), daemon=True).start()  
 app = wx.App(False)  
 self.frame = MyFrame(self.graphic\_q)  
 self.frame.Show()  
 app.MainLoop()  
  
  
 def monitor\_file(self, file\_path: str, end\_q: Queue):  
 *"""  
 monitors a file and sends server what changed in file  
 :param file\_path: path for file to monitor  
 :param end\_q: queue to kill monitoring  
 :return: None  
 """* modified = 0x00000003  
 FILE\_LIST\_DIRECTORY = 0x0001  
 FILE\_NOTIFY\_CHANGE\_FILE\_NAME = 0x0001  
 FILE\_NOTIFY\_CHANGE\_LAST\_WRITE = 0x0010  
 FILE\_FLAG\_BACKUP\_SEMANTICS = 0x02000000  
 OPEN\_EXISTING = 3  
  
 path\_to\_watch, file\_name = FileHandler.split\_path\_last\_part(file\_path)  
 directory\_handle = win32file.CreateFileW(  
 path\_to\_watch,  
 FILE\_LIST\_DIRECTORY, *# No access (required for directories)* win32file.FILE\_SHARE\_READ | win32file.FILE\_SHARE\_WRITE | win32file.FILE\_SHARE\_DELETE,  
 None,  
 OPEN\_EXISTING,  
 FILE\_FLAG\_BACKUP\_SEMANTICS,  
 None  
 )  
 if directory\_handle == -1:  
 print("Error opening directory")  
 else:  
 *# changed = False* while True:  
 if not end\_q.empty():  
 break  
  
 try:  
 result = win32file.ReadDirectoryChangesW(  
 directory\_handle,  
 4096,  
 True, *# Watch subtree* FILE\_NOTIFY\_CHANGE\_LAST\_WRITE | FILE\_NOTIFY\_CHANGE\_FILE\_NAME,  
 None  
 )  
  
 for action, name\_monitored in result:  
  
 if action == modified and name\_monitored == file\_name:  
 print('saved file in monitor')  
 to\_send\_path = file\_path.replace(Settings.tmp\_directory\_name, '', 1)  
 self.save\_file(file\_path, to\_send\_path)  
  
 except Exception:  
 break  
  
 def save\_file(self, from\_path, to\_path):  
 *"""  
 sends file data to ip inside path, if local saves path local  
 :param from\_path: location of data to send  
 :param to\_path: the path for client to save data in  
 :return: None  
 """* if not os.path.isfile(from\_path):  
 self.call\_error('could not save file data')  
 return  
  
 *# get file data* with open(from\_path, 'rb') as f:  
 file\_data = f.read()  
  
 if self.file\_handler.is\_local(to\_path):  
 to\_path = self.file\_handler.remove\_ip(self.user\_name, to\_path)  
 with open(to\_path, 'wb') as f:  
 f.write(file\_data)  
  
 *# need to send to other computer* else:  
 ip\_to\_send = FileHandler.extract\_ip(self.user\_name, to\_path)  
 if ip\_to\_send in self.ip\_comm:  
 comm = self.ip\_comm[ip\_to\_send]  
 else:  
 rcv\_q = Queue()  
 comm = ClientComm(ip\_to\_send, Settings.pear\_port, rcv\_q, 8, 'U')  
 self.ip\_comm[ip\_to\_send] = comm  
 threading.Thread(target=self.rcv\_comm, args=(comm,), daemon=True).start()  
  
 header = protocol.pack\_change\_file(to\_path)  
 comm.send\_file(header, file\_data)  
  
 def visualize\_open\_file(self, file\_path, comm: ClientComm):  
 *"""  
 opening file to user and activates monitoring on file  
 :param file\_path: path of file  
 :param comm: the client\_comm opened with other pc  
 :return: None  
 """  
  
 # start monitoring file* q = Queue()  
 threading.Thread(target=self.monitor\_file, args=(file\_path, q), daemon=True).start()  
  
 process\_name = process\_handler.get\_process\_name(file\_path)  
 ls1 = process\_handler.get\_all\_pid(process\_name)  
 dir\_path, \_ = FileHandler.split\_path\_last\_part(file\_path)  
 FileHandler.open\_file(file\_path)  
 self.refresh\_cursor()  
 print('ls1:', ls1)  
 *# wait for file to be added to pid list* while True:  
 ls2 = process\_handler.get\_all\_pid(process\_name)  
 print('ls2:', ls2)  
 if ls2 != ls1:  
 break  
  
 new\_pid = set(ls2) - set(ls1)  
 pid = list(new\_pid)[0]  
 print('this is pid:', pid)  
 process\_handler.wait\_for\_process\_to\_close(pid)  
  
 *# closing all related connections to pc* ip = next(filter(lambda item: item[1] == comm, self.ip\_comm.items()), None)[0]  
 del self.ip\_comm[ip]  
 comm.close()  
  
 *# generic action on file to end thread of monitoring file* q.put('end')  
 FileHandler.delete(file\_path)  
  
 print('ended visualize file')  
  
 def rcv\_graphic(self, q: Queue):  
 *"""  
 gets q from graphics and handles what got  
 :param q: q gets from graphic  
 :return: None  
 """* while True:  
 print(self.file\_tree)  
 command, param\_got = q.get()  
 print('command:', command, ' param:', param\_got)  
 if command == 'create file':  
 path, name = self.file\_handler.split\_path\_last\_part(param\_got)  
 name = name.split('.')  
 typ = name[-1]  
 name = ''.join(name[:-1])  
 self.create(path, name, typ)  
 elif command == 'create folder':  
 path, name = self.file\_handler.split\_path\_last\_part(param\_got)  
 self.create(path, name, 'fld')  
  
 elif command == 'open':  
 self.get\_file\_data(param\_got)  
  
 elif command == 'delete':  
 self.delete(param\_got)  
  
 elif command == 'rename':  
 path, new\_name = self.file\_handler.split\_path\_last\_part(param\_got)  
 print('path:', path)  
 print('new name:', new\_name)  
 self.rename(path, new\_name)  
  
 elif command == 'login':  
 if param\_got.count(',') != 1:  
 self.call\_error('problem with password or username')  
 continue  
 self.user\_name, password = param\_got.split(',')  
 print('self.username:', self.user\_name)  
 self.do\_connect(password)  
  
 elif command == 'register':  
 if param\_got.count(',') != 1:  
 self.call\_error('problem with password or username')  
 continue  
 self.user\_name, password = param\_got.split(',')  
 print('self.username:', self.user\_name)  
 self.do\_register(password)  
  
 elif command == 'upload file':  
 if param\_got.count(',') != 1:  
 self.call\_error('problem with uploading file')  
 continue  
 dir\_path, from\_path = param\_got.split(',')  
 name = os.path.basename(from\_path)  
 to\_path = f'{dir\_path}\\{name}'  
 name, typ = FileHandler.split\_name\_typ(name)  
  
 self.handle\_status\_create(['ok', f'{dir\_path}\\{name}', typ])  
 self.save\_file(from\_path, to\_path)  
  
 elif command == 'download':  
 if param\_got.count(',') != 1:  
 self.call\_error('problem with uploading file')  
 continue  
 downloaded, download\_to = param\_got.split(',')  
  
 print('download:', downloaded)  
 dir\_path, name = FileHandler.split\_path\_last\_part(download\_to)  
 typ = os.path.basename(downloaded).split('.')[-1]  
 FileHandler.create(f'{dir\_path}\\{name}', str(typ))  
 if '.' not in os.path.basename(download\_to):  
 download\_to += f'.{typ}'  
 self.downloads[download\_to] = downloaded  
 print('local path:', download\_to)  
 self.get\_file\_data(downloaded)  
  
 elif command == 'paste':  
 if param\_got.count(',') != 1:  
 self.call\_error('problem with uploading file')  
 continue  
 file\_to\_copy, copy\_to = param\_got.split(',')  
 self.clone\_file(file\_to\_copy, copy\_to)  
  
 elif command == 'move':  
 if param\_got.count(',') != 1:  
 self.call\_error('problem with uploading file')  
 continue  
 file\_to\_move, move\_to = param\_got.split(',')  
 self.move\_file(file\_to\_move, move\_to)  
  
 else:  
 print('wrong output')  
  
  
  
 def move\_file(self, file\_to\_move: str, move\_to: str):  
 *"""  
 moves file from one folder to another  
 :param file\_to\_move: full path of the file to move  
 :param move\_to: the path to move the file to  
 :return: None  
 """  
 # if got a file path changing it to the folder* if move\_to not in self.file\_tree:  
 move\_to, \_ = FileHandler.split\_path\_last\_part(move\_to)  
  
 file\_dir\_path, file\_name = FileHandler.split\_path\_last\_part(file\_to\_move)  
  
 *# if moving to same dir* if move\_to == file\_dir\_path or file\_to\_move in self.file\_tree:  
 self.call\_error('can not move a folder only files')  
 return  
  
 print('move\_to:', move\_to)  
 print('file\_to\_move:', file\_to\_move)  
  
 file\_name = FileHandler.build\_name\_for\_file(self.file\_tree, move\_to, file\_to\_move, '(moved)')  
 new\_file\_path = str(os.path.join(move\_to, file\_name))  
  
 if self.file\_handler.is\_local(move\_to) and self.file\_handler.is\_local(file\_to\_move):  
 *# making folders local* local\_new\_file\_path = FileHandler.remove\_ip(self.user\_name, new\_file\_path)  
 local\_file\_to\_move = FileHandler.remove\_ip(self.user\_name, file\_to\_move)  
  
 moved = FileHandler.move(local\_file\_to\_move, local\_new\_file\_path)  
 if moved:  
 print('removing:', file\_to\_move)  
 self.folders\_remove(file\_to\_move)  
 print(f'adding: {move\_to}\\{file\_name}')  
 file\_name, typ = FileHandler.split\_name\_typ(file\_name)  
 self.folders\_add(move\_to, file\_name, typ)  
 self.folders\_remove(file\_to\_move)  
 else:  
 self.call\_error(f'could not move {file\_name}')  
  
 else:  
 self.client.send(protocol.pack\_do\_move(file\_to\_move, new\_file\_path))  
  
  
 def clone\_file(self, file\_to\_copy: str, copy\_to: str):  
 *"""  
 copies file from one folder to another  
 :param file\_to\_copy: full path of the file to copy  
 :param copy\_to: the path to copy the file to  
 :return: None  
 """  
 # if got a file path changing it to the folder* if copy\_to not in self.file\_tree:  
 copy\_to, \_ = FileHandler.split\_path\_last\_part(copy\_to)  
  
 file\_name = FileHandler.build\_name\_for\_file(self.file\_tree, copy\_to, file\_to\_copy, '(copy)')  
 new\_file\_path = str(os.path.join(copy\_to, file\_name))  
  
 if self.file\_handler.is\_local(copy\_to) and self.file\_handler.is\_local(file\_to\_copy):  
 *# making folders local* local\_new\_file\_path = FileHandler.remove\_ip(self.user\_name, new\_file\_path)  
 local\_file\_to\_copy = FileHandler.remove\_ip(self.user\_name, file\_to\_copy)  
  
 copied = FileHandler.direct\_copy\_file(local\_file\_to\_copy, local\_new\_file\_path)  
 if copied:  
 file\_name, typ = FileHandler.split\_name\_typ(file\_name)  
 self.folders\_add(copy\_to, file\_name, typ)  
 else:  
 self.call\_error(f'could not copy {file\_name}')  
  
 else:  
 self.client.send(protocol.pack\_do\_clone(file\_to\_copy, new\_file\_path))  
  
  
 def handle\_status\_rename(self, vars: list):  
 *"""  
 gets status of renaming and shows user what happened  
 :param vars: status, location, new\_name  
 :return: None  
 """* status, location, new\_name = vars  
  
 if status == 'ok':  
 self.folders\_rename(location, new\_name)  
 else:  
 self.call\_error(f'could not rename to "{new\_name}"')  
  
  
 def handle\_status\_clone(self, vars: list):  
 *"""  
 gets status of cloning and shows user what happened  
 :param vars: status, old\_location, new\_location  
 :return: None  
 """* status, copy\_from, copy\_to = vars  
 print('copy\_from:', copy\_from)  
 print('copy\_to:', copy\_to)  
 folder\_copied\_to, file\_name = FileHandler.split\_path\_last\_part(copy\_to)  
 just\_name, typ = FileHandler.split\_name\_typ(file\_name)  
  
 *# informing graphics* if status == 'ok':  
 self.folders\_add(folder\_copied\_to, just\_name, typ)  
 else:  
 self.call\_error(f'could not clone {just\_name}.{typ}')  
  
  
 def handle\_status\_move(self, vars: list):  
 *"""  
 gets status of moving and shows user what happened  
 :param vars: status, old\_location, new\_location  
 :return: None  
 """* status, move\_from, move\_to = vars  
 folder\_copied\_to, file\_name = FileHandler.split\_path\_last\_part(move\_to)  
  
 just\_name, typ = FileHandler.split\_name\_typ(file\_name)  
  
 *# informing graphics* if status == 'ok':  
 self.folders\_add(folder\_copied\_to, just\_name, typ)  
 self.folders\_remove(move\_from)  
 else:  
 self.call\_error(f'could not clone {just\_name}.{typ}')  
  
 def rename(self, path: str, new\_name: str):  
 *"""  
 gets path and typ, creates it  
 :param path: path of object to create  
 :param new\_name: the new name of the file  
 :return: None  
 """  
 # trying to change computers folder* if path in self.file\_tree and '.' in os.path.basename(path):  
 self.call\_error('cannot rename computers directory')  
 return  
  
 if self.file\_handler.is\_local(path):  
 local = self.file\_handler.remove\_ip(self.user\_name, path)  
  
 *# do local stuff of creating file* if self.file\_handler.rename(local, new\_name):  
 self.folders\_rename(path, new\_name)  
  
 else:  
 self.call\_error(f'could not rename to "{new\_name}"')  
 else:  
 self.client.send(protocol.pack\_do\_rename(path, new\_name))  
  
 def call\_error(self, error: str):  
 *"""  
 gets error and sends it to graphic  
 :param error: error to show user  
 """* self.send\_to\_graphics('error', {'error': error})  
  
 def delete(self, path: str):  
 *"""  
 gets path of file with its name and deletes it  
 :param path: path of file  
 return: None  
 """  
 # trying to change computers folder* if path in self.file\_tree and '.' in os.path.basename(path):  
 self.call\_error('cannot delete computers directory')  
 return  
  
 if self.file\_handler.is\_local(path):  
 local = self.file\_handler.remove\_ip(self.user\_name, path)  
  
 *# do local stuff deleting file* self.file\_handler.delete(local)  
 self.folders\_remove(path)  
 else:  
 self.client.send(protocol.pack\_do\_delete(path))  
  
 def rcv\_comm(self, comm: ClientComm):  
 *"""  
 gets data from server and calls functions accordingly  
 :param : client comm  
 """* q = comm.rcv\_q  
 commands = {'02': self.handle\_status\_register, '04': self.handle\_status\_login, '05': self.handle\_got\_file\_tree,  
 '09': self.handle\_status\_rename, '07': self.handle\_status\_create, '13': self.handle\_status\_move,  
 '15': self.handle\_status\_clone, '17': partial(self.handle\_status\_open, comm),  
 '11': self.handle\_status\_delete, '19': self.handle\_status\_saved\_file}  
 while True:  
 data = protocol.unpack(q.get())  
  
 if not data:  
 print('got None from protocol')  
 continue  
  
 print('data from server:', data)  
 opcode, params = data  
 if opcode == '00':  
 print('quit')  
 break  
 commands[opcode](params)  
  
 def get\_file\_tree(self):  
 *"""  
 gets file tree and lets user navigate through it  
 :return: None  
 """* print('user\_path:', self.file\_handler.user\_path)  
 user\_path = self.file\_handler.user\_path[:-1]  
  
 while True:  
 new\_folders = {}  
 *# gets new folder adds it to folders dict* folders\_got = self.handle\_tree.get()  
 new\_folders[user\_path] = [',']  
 self.file\_tree.update(folders\_got)  
  
 *# gets ip of computer got from* ip\_path = list(folders\_got.keys())[0]  
 ip = os.path.basename(ip\_path)  
  
 *# adds ip to folders in path needed* self.file\_tree[user\_path].insert(0, ip)  
 new\_folders.update(folders\_got)  
 new\_folders[user\_path].insert(0, ip)  
 print('new\_folders:', new\_folders)  
 self.send\_to\_graphics('update\_tree', {'dic': new\_folders})  
  
 def handle\_status\_saved\_file(self, vars: list):  
 *"""  
 calling error if was a problem saving file  
 :param vars: status and path  
 :return: None  
 """* status = (vars[0] == 'ok')  
 if len(vars) != 2:  
 self.call\_error(f"couldn't save file, problem communicating with other pc")  
 return  
 path = vars[1]  
 if not status:  
 self.call\_error(f"couldn't save file: {path}")  
  
 def handle\_status\_open(self, comm: ClientComm, vars: list):  
 *"""  
 showing user the file if opened  
 :param vars: status and path  
 :return: None  
 """* status = vars[0]  
 if status == 'no':  
 self.call\_error('could not open file')  
 return  
 else:  
  
 path = vars[1]  
 print('path:', path)  
 download\_path = path.replace(Settings.tmp\_directory\_name, '', 1)  
 print('downloaded\_path:', download\_path)  
 print('downloads:', self.downloads)  
  
 if download\_path in self.downloads.values():  
 local\_path = next(filter(lambda item: item[1] == download\_path, self.downloads.items()), None)[0]  
 FileHandler.direct\_copy\_file(path, local\_path)  
 del self.downloads[local\_path]  
 self.refresh\_cursor()  
  
 elif os.path.isfile(path):  
 threading.Thread(target=self.visualize\_open\_file, args=(path, comm), daemon=True).start()  
 else:  
 self.call\_error('error in opening file')  
  
  
 def get\_file\_data(self, path: str):  
 *"""  
 gets path of file with its name and opens it  
 :param path: path of file  
 return:  
 """* print('path of file to get data:', path)  
 *# file is local so opens it immediately* if self.file\_handler.is\_local(path):  
 local = self.file\_handler.remove\_ip(self.user\_name, path)  
 FileHandler.open\_file(local)  
 self.refresh\_cursor()  
  
 *# if user asked to download local file* elif path in self.downloads and os.path.isfile(FileHandler.remove\_ip(self.user\_name, self.downloads[path])):  
 download\_from\_path = FileHandler.remove\_ip(self.user\_name, self.downloads[path])  
 print('download from:', download\_from\_path)  
 print('download to:', path)  
 FileHandler.direct\_copy\_file(download\_from\_path, path)  
 del self.downloads[path]  
 self.refresh\_cursor()  
  
 *# if file is on another computer* else:  
 ip\_to\_connect = self.file\_handler.extract\_ip(self.user\_name, path)  
 *# setting up pear to pear* if ip\_to\_connect not in self.ip\_comm:  
 rcv\_q = Queue()  
  
 comm = ClientComm(ip\_to\_connect, Settings.pear\_port, rcv\_q, 8, 'U')  
 self.ip\_comm[ip\_to\_connect] = comm  
 threading.Thread(target=self.rcv\_comm, args=(comm,), daemon=True).start()  
  
 else:  
 comm = self.ip\_comm[ip\_to\_connect]  
  
 comm.send(protocol.pack\_do\_open\_file(path))  
  
 def folders\_remove(self, path):  
 *"""  
 gets path and removes it from folders and tells graphic  
 :param path: path to delete  
 """* father\_path, name = self.file\_handler.split\_path\_last\_part(path)  
 if path in self.file\_tree:  
 del self.file\_tree[path]  
  
 if father\_path in self.file\_tree and name in self.file\_tree[father\_path]:  
 self.file\_tree[father\_path].remove(name)  
  
 self.send\_to\_graphics('delete', {'path': path})  
  
 def folders\_add(self, path, name, typ):  
 *"""  
 gets path and adds it to folders dict  
 :param path: path of father  
 :param name: name of file  
 :param typ: type of file/folder  
 """* if path.endswith('\\'):  
 path = path[:-1]  
 if typ == 'fld':  
 self.file\_tree[f'{path}\\{name}'] = [',']  
 self.file\_tree[path].insert(0, name)  
  
 else:  
 self.file\_tree[path].append(f'{name}.{typ}')  
  
 self.send\_to\_graphics('create', {'path': path, 'name': name, 'typ': typ})  
  
 def folders\_rename(self, old\_path: str, new\_name: str):  
 *"""  
 tells graphics to rename object  
 :param old\_path: the old path of the object  
 :param new\_name: the new name of the object  
 :return None  
 """* folder\_path, old\_name = FileHandler.split\_path\_last\_part(old\_path)  
 new\_path = os.path.join(folder\_path, new\_name)  
  
 *# folder* if '.' not in new\_name:  
 self.file\_tree[new\_path] = self.file\_tree.pop(old\_path)  
  
 *# file* else:  
 self.file\_tree[folder\_path].remove(old\_name)  
 self.file\_tree[folder\_path].append(new\_name)  
  
 self.send\_to\_graphics('rename', {'old\_path': old\_path, 'new\_name': new\_name})  
 *# wx.CallAfter(pub.sendMessage, 'rename', old\_path=old\_path, new\_name=new\_name)* def refresh\_cursor(self):  
 *"""  
 refreshes cursor at graphics  
 """* wx.CallAfter(pub.sendMessage, 'cursor')  
  
 def send\_to\_graphics(self, flag: str, params: dict):  
 *"""  
 gives graphics a command to do  
 :param flag: the flag of  
 :param params: the variables to pass to the graphics  
 """* wx.CallAfter(pub.sendMessage, flag, \*\*params)  
 self.refresh\_cursor()  
  
 def create(self, path: str, name: str, typ: str):  
 *"""  
 gets path and typ, creates it  
 :param path: path of object to create  
 :param typ: type of object  
 :param name: name of object  
 :return: None  
 """* if not path.endswith('\\'):  
 path += '\\'  
 if self.file\_handler.is\_local(path):  
 local = self.file\_handler.remove\_ip(self.user\_name, path)  
 local = local + name  
  
 *# do local stuff of creating file* self.file\_handler.create(local, typ)  
 self.folders\_add(path, name, typ)  
 else:  
 self.client.send(protocol.pack\_do\_create(path + name, typ))  
  
 def handle\_status\_create(self, vars: list):  
 *"""  
 gets status of creation and shows user what happened  
 :param vars: status, location, type  
 :return: None  
 """* status, location, typ = vars  
 if status == 'ok':  
 path, name = FileHandler.split\_path\_last\_part(location)  
 self.folders\_add(path, name, typ)  
  
 else:  
 self.call\_error('could not create object')  
  
 def handle\_status\_delete(self, vars: list):  
 *"""  
 gets status of creation and shows user what happened  
 :param vars: status, location, type  
 :return: None  
 """* status, location = vars  
 if status == 'ok':  
 self.folders\_remove(location)  
  
 else:  
 self.call\_error(f'could not delete {os.path.basename(location)}')  
  
 def handle\_status\_register(self, vars):  
 *"""  
 gets status and shows user what happened  
 :param vars: success or failure  
 :return: None  
 """* success = vars[0]  
 if success == 'ok':  
 self.send\_to\_graphics('register\_ok', {})  
 else:  
 self.call\_error('username already exists')  
  
 def handle\_status\_login(self, vars):  
 *"""  
 gets status, shows user what happened also creates hidden root directory if success and starts navigate folder  
 :param vars: success or failure  
 :return: None  
 """* success = vars[0]  
 if success == 'ok':  
 print('you are signed in')  
  
 self.file\_handler = FileHandler(self.user\_name)  
 user\_path = self.file\_handler.user\_path[:-1]  
 self.file\_tree = {user\_path: [',']}  
 self.file\_handler.create\_root()  
 print('user\_path:', user\_path)  
  
 *# thread that waits for getting file trees* threading.Thread(target=self.get\_file\_tree, daemon=True).start()  
 *# grpahics* self.send\_to\_graphics('login', {})  
  
 else:  
 self.call\_error('username or password does not exist')  
 print('could not sign in')  
  
 def handle\_got\_file\_tree(self, vars):  
 *"""  
 gets file tree and start interaction with user about files  
 :param vars: file tree  
 :return: None  
 """* self.handle\_tree.put(vars[0])  
  
 def do\_register(self, password):  
 *"""  
 gets username and password and sends it to server by protocol to register  
 :param password: password  
 :return: None  
 """* to\_send = protocol.pack\_register(self.user\_name, password)  
 self.client.send(to\_send)  
  
 def do\_connect(self, password):  
 *"""  
 gets username and password and sends it to server by protocol to sign in  
 :param password: password  
 :return: None  
 """* to\_send = protocol.pack\_sign\_in(self.user\_name, password, Settings.get\_mac\_address())  
 self.client.send(to\_send)  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 MainUserClient()

**נספח א – החקר שבוצע בפרויקט**

**במהלך הפרויקט נתקלתי בבעיה, כאשר התעסקתי עם הפרויקט ללא לכבות אותו לאורך זמן כל הפעולות התחילו לקחת יותר ויותר זמן עד שלפעמים שינוי שם קובץ היה לוקח יותר מ10 שניות. התחלתי לחפש את שורש העניין וגיליתי שכאשר אני פותח קובץ, למרות שאני סוגר אותו יש תהליכונים שעדיין לא נסגרים, כאשר הבנתי זאת הגעתי למקום בו אני יוצר את התהליכון שאחראי על ניתור הקובץ ולמרות השערתי גיליתי כי הוא אמור להיסגר. כאשר יצרתי את התהליכון הוא היה מוגדר שכאשר האבא של התהליכון נסגר גם התהליכון ייסגר מיידית (daemon=true). התחלתי להתעסק טיפה עם התהליכונים כדי עדיין להבין למה התהליכון לא נסגר למרות שהאבא שלו הסתיים. גיליתי שכאשר אני יוצר את תהליכון ב' מתוך תהליכון א' אין זה אומר שכאשר תהליכון א' מסתיים גם תהליכון ב' ייסגר אלא רק כאשר התהליך שיצר את תהליכון א' ייסגר רק אז תהליכון ב' ייסגר.**