**GolDrive – תוכנה** לשמירת קבצים



שם המגיש: ריף גולד  
תעודת זהות: 330621434  
בית ספר ועיר: עתיד כרמים בנימינה  
מורה: מרי גבע  
מועד הגשה: 5.2024

**תוכן עניינים**

[**פרק א - 'goldrive' - ייזום** 3](#_Toc166752842)

[**פרק ב - 'goldrive' - אפיון** 5](#_Toc166752843)

[**פרק ג - 'goldrive' - ניתוח** 10](#_Toc166752844)

[**פרק ד - 'goldrive' - עיצוב** 13](#_Toc166752845)

[**פרק ה - 'goldrive' – הקוד** 63](#_Toc166752846)

[**פרק ו - 'goldrive' – בדיקות('קופסא לבנה')** 68](#_Toc166752847)

[**פרק ז - 'goldrive' – מדריך למשתמש** 70](#_Toc166752848)

[**פרק ח - 'goldrive' – רפלקציה** 72](#_Toc166752849)

[**פרק ט - 'goldrive' – ביבליוגרפיה** 73](#_Toc166752850)

[**נספחים** 74](#_Toc166752851)

# **פרק א - 'goldrive' - ייזום**

**1.** הפרויקט הוא תוכנת שיתוף. לכל משתמש יהיה אזור אישי ואזור שיתופי, באזור האישי הקבצים נגישים רק אל המשתמש ובאזור השיתופי המשתמש יוכל לשתף קבצים וגם הקבצים ששותפו אתו ימצאו שם. קבצים יוכלו להינעל על ידי סיסמא כדי ליצור עוד שכבה של פרטיות לקבצים פרטיים. בחרתי בפרויקט משום שהוא היה נראה מעניין ויכול לתרום לי בכך שישמור לי קבצים במקום שאני סומך עליו ואני יודע שהוא מאובטח. אתגרים שצפויים לי בפרויקט היא אחסון הקבצים בצורה יעילה ושיתוף הקבצים בין משתמשים.

**2.** הפרויקט מיועד לאנשים שמשתמשים בהרבה קבצים ובעיקר לאנשים שמשתפים קבצים עם אנשים אחרים, משום שאפשר בנוחות לאחסן קבצים משותפים ואישיים במקום אחד.

**3.** מטרת הפרויקט היא:

* לאפשר למשתמשים מקום אחסון בטוח ונוח לשימוש לקבצים חשובים או קבצי עבודה ופנאי.
* לאפשר מקום נגיש לכל משתמש רשום מכל מקום ברשת.
* ליצור דרך נוחה ופשוטה לשיתוף קבצים עם משתמשים אחרים.

**4.** הבעיה והתועלות:  
הבעיה העיקרית הינה מציאת פתרון ליכולת מצד אחד לשמור קבצים בצורה מאובטחת ומהצד השני אפשרות לגשת לקבצים האלו מכל מקום ברשת, לשתפם עם אחרים ועוד.  
התועלות של המערכת שלי היא במתן מקום אחסון נוח ומאובטח ברשת עם יכולת לגשת אליו בדרך בטוחה ועם יכולות של שיתוף קבצים עם אחרים ועוד.

סקירת פתרונות הקיימים בשוק:

פתרון ראשון שקיים בשוק הוא הגוגל דרייב, בעזרת חשבון הGmail שלך ניתן לגשת לדרייב ולשמור קבצים שם, הקבצים נשמרים אוטומטית, הבעיה עם הדרייב של גוגל הוא שצריך משתמש מייל ולא מתאפשרת נעילת קבצים.   
הפתרון השני שקיים הוא דרופבוקס, בדומה לדרייב של גוגל התוכנה היא מקום שבו ניתן לשמור קבצים במקום מרוכז אחד, וגם היא לא מאפשרת לנעול קבצים. בנוסף אינה שומרת את הקבצים בצורה אוטומטית לאחר שעורכים אותם.

**5.** הטכנולוגיה אינה חדשה ונמצאת בשוק כבר מעל ל15 שנה אך היא כל הזמן משתנה ומשתפרת, בין אם זה ביעילות השמירה של הקבצים או ההצפנה.

הגבלה שתוכל לנבוע היא הבעיה של שמירת הקבצים משום שייתכן שייגמר המקום במקום שבו אנחנו שומרים את הקבצים.  
בנוסף בגלל הגבלות הרשת של בית הספר, התוכנה תהיה סגורה ותתאפשר גישה אך ורק למשתמשים ברשת הבית ספרית.

**6.** התחומים בהם הפרויקט עוסק הוא רשתות (שרת מרובה לקוחות), משום שהעלאות קבצים והורדת קבצים מהשרת משתמשת ברשת, קבצים, מערכות הפעלה משום שיש הרבה threadים וכמובן גם הצפנה חזקה כדי לשמור על פרטיות הקבצים.

**7.** הפרויקט מתחבר לקיימות בכך ששומר על עבודה ממוחשבת ובכך חוסך כריתת עצים לייצור ניירות משום שהקבצים נמצאים במחשב ולא נכתבים על דף.

# **פרק ב - 'goldrive' - אפיון**

**1.** הפרויקט יהיה מקום לשמור קבצים בצורה נוחה ויעילה הוא יאפשר למשתמשיו גם נוחות וגם ביטחון משום שקבציי הפרויקט יהיו מוצפנים וכל משתמש יהיה מוגן בעזרת שם משתמש וסיסמא, בנוסף לכך לכל קובץ תהיה אפשרות נעילה שתספק מנגנון הגנה. והכי חשוב ניתן יהיה לגשת לקבצים מכל מחשב ברשת.

**2.** הפרויקט יעניק את היכולות הבאות**:**

* הרשמה כמשתמש חדש במערכת
* התחברות למשתמש קיים במערכת
* העלאת קבצים לאזור הבטוח
* הורדת קבצים מהאזור הבטוח
* שיתוף קבצים עם חברים אחרים
* מחיקת קבצים
* נעילת קבצים
* פעולות שונות על הקבצים כגון שכפול, שינוי שם ועוד
* גישה לאזור הבטוח מכל מקום ברשת
* אבטחה מרבית לקבצים (הצפנה, נעילה, סיסמאות ועוד)

**3. בדיקות קופסא שחורה:**

| **מספר** | **שם הבדיקה (שם שיעיד על תוכן הבדיקה)** | **מה אמורה לבדוק** | **איך מתכננים לבדוק(לתאר בפירוט את שלבי הבדיקה)** |
| --- | --- | --- | --- |
| **1** | **התחברות** | **האם אפשר להתחבר למשתמש כמו שצריך** | **לפתוח את המערכת, להתחבר עם שם משתמש וסיסמא נכונים ולראות שמצליחים להיכנס למערכת.**  **לחזור על הפעולה עם שם משתמש וסיסמא שגויה ולראות שלא מצליחים להיכנס למערכת** |
| 2 | העלאת קבצים | האם אפשר לשמור קבצים למשתמש | לפתוח את המערכת ולהתחבר עם שם סיסמא ומשתמש, לנסות להעלות קובץ ולראות שהוא עלה ומוצג בקבצים שלי, במידה והוא עלה להתנתק מהמשתמש ולהתחבר שוב ולראות שהקובץ עדיין מוצג. |
| 3 | הורדת קבצים | האם אפשר להוריד קבצים | לפתוח את התוכנה, להתחבר למשתמש, לבחור קובץ ולנסות להוריד אותו, לבדוק האם הקובץ מורד כמו שצריך ובמקום הנכון. |
| 4 | איש שלישי | האם אפשר לקרוא מידע שעובר | לפתוח Wireshark ואז לפתוח את התוכנה, להתחבר למשתמש ולנסות להעלות קובץ אחר כך לבדוק Wireshark האם אפשר לקרוא את המידע שעבר או שהוא מוצפן. |
| 5 | שיתוף קבצים | האם אפשר לשתף קובץ עם חבר | לפתוח את התוכנה להתחבר למשתמש, לבחור קובץ ולשתף אותו עם משתמש אחר, להתחבר למשתמש האחר ולראות האם הקובץ ששותף מופיע באזור השיתופים. |
| 6 | מחיקת קבצים | האם אפשר למחוק קבצים | לפתוח את התוכנה, להתחבר למשתמש, לבחור קובץ ולנסות למחוק אותו, לראות שהקובץ נמחק ולא מופיע, להתחבר מחדש למשתמש ולראות שהקובץ עדיין לא מופיע. |
| 7 | שינוי שם | האם אפשר לשנות שם לקובץ | לפתוח את התוכנה, להתחבר למשתמש, לבחור קובץ ולנסות לשנות לו את השם, לראות שהשם משתנה כמו שצריך, להתחבר מחדש ולראות שהשם נותר משונה. |
| 8 | כניסה כפולה | לבדוק האם המערכת מונעת ממשתמש אחד להיכנס בו זמנית ממחשבים שונים | לפתוח את התוכנה, ליצור משתמש, לראות שמתחבר כמו שצריך, ולנסות להתחבר לאותו משתמש ממחשב אחר ולראות שזה **לא** מתחבר. |
| 9 | בדיקת ביצועים | לבדוק האם המערכת מבצעת פעולות במהירות | לפתוח את התוכנה, להתחבר למשתמש ולנסות לבדוק כל פעולה שמתאפשרת לקובץ ולבדוק שהכל נעשה במהירות. |
| 10 | בדיקת יציבות | לבדוק האם המערכת מתפקדת בקביעות למרות שיש מספר משתמשים מחוברים | לפתוח את התוכנה במספר מחשבים, להתחבר מכל מחשב למשתמש ומכל משתמש להריץ איזושהי פעולה על קובץ, לראות שבכל משתמש הפעולה נעשתה כמו שצריך. |
| 11 | זמני תגובה | לבדוק האם יש הבדלים בזמני התגובה של המערכת כאשר מורידים/מעלים קבצים בגדלים שונים | לפתוח את התוכנה, להתחבר למשתמש ולנסות להעלות ולהוריד קבצים שונים בעלי גודל שונה ולבדוק את הזמן שלוקח. |

**4. תכנון לו"ז:**

| **פעילות** | **זמן התחלה מתוכנן** | **זמן סיום מתוכנן** | **זמן התחלה בפועל** | **זמן סיום בפועל** | **הערות** |
| --- | --- | --- | --- | --- | --- |
| יזום | 1.11.23 | 23.11.23 | 6.11.23 | 7.11.23 | הוגש לפני הזמן |
| אפיון | 23.11.23 | 1.12.23 | 14.11.23 | 18.11.23 | הוגש לפני הזמן |
| ניתוח | 1.12.23 | 15.12.23 | 18.11.23 | 30.11.23 | הוגש לפני הזמן |
| עיצוב | 15.12.23 | 15.1.24 | 30.11.23 | 11.1.24 | הוגש לפני הזמן |
| בניית שרת מרובה לקוחות בסיסי | 15.1.24 | 18.1.24 | 11.1.24 | 11.1.24 | סוים לפני הזמן |
| הוספת ממסד נתונים | 18.1.24 | 20.1.24 | 11.1.24 | 11.1.24 | סוים לפני הזמן |
| בניית ממשק לקוח | 20.1.24 | 30.1.24 | 12.1.24 | 22.1.24 | סוים לפני הזמן |
| טיפול בקבצים העלאת קבצים מחיקת קבצים שינוי שם | 30.1.24 | 10.2.24 | 25.1.24 | 26.1.24 | סוים לפני הזמן |
| בניית יכולת לשיתוף קבצים עם משתמשים אחרים | 10.2.24 | 16.2.24 | 28.1.24 | 28.1.24 | סוים לפני הזמן |
| הוספת הצפנות | 16.2.24 | 18.2.24 | 11.1.24 | 12.1.24 | סוים לפני הזמן |
| תיקון באגים | 18.2.24 | 25.2.24 | תוך כדי הפרויקט | תוך כדי הפרויקט | התבצע תוך כדי כתיבת הקוד |
| גרסה ראשונית | 25.2.24 | 1.3.24 | 11.1.24 | 1.3.24 | הוגש בזמן, הקוד כבר היה כתוב מעל 95% ולא רק 80% |
| מסמך בדיקות | 1.3.24 | 10.4.24 | 12.5.24 | 12.5.24 | התעכבתי בכתיבת המסמכים, הוגש באיחור |
| מדריך למשתמש | 10.4.24 | 1.5.24 |  |  |  |
| גרסה סופית | 10.5.24 | 10.5.24 |  |  |  |
| סגירת תיק פרויקט |  |  |  |  |  |
| מצגת הסבר |  |  |  |  |  |

**5. סיכונים:**

| **הסיכון** | **פירוט הסיכון** | **רמת הסיכון**  **(קל/בינוני/קשה)** | **תיאור דרכים (לפחות 2 ) להתמודדות עם הסיכון ולהקטין אותו** | **מה בוצע בפועל** | **תאריך** |
| --- | --- | --- | --- | --- | --- |
| תכנות מבולגן | ייקח יותר זמן לתכנת כל דבר | קל | * מההתחלה ליצור פעולות ומחלקות נוחות * לכתוב קוד קריא וברור להבנה | הקוד היה כתוב יפה ומסודר | 2.5.24 |
| להתקע על פרטים קטנים | פרויקט יוגש באיחור | בינוני | * גם אם יש באג קטן להמשיך לאבנים הגדולות ולחזור אחר כך * לא קל דבר קטן צריך לעבוד עליו ואפשר לפעמים לוותר על דברים | לא נתקעתי על פרטים קטנים יותר מידי | 2.5.24 |
| תכנות לא יעיל | ירדו נקודות | בינוני | * להיעזר בחברים כדי לקבל עצות * להיעזר באינטרנט כדי למצוא פתרונות יעילים של אנשים לבעיות | התכנות כתוב בצורה יעילה | 2.5.24 |
| אי עמידה בזמנים | פרויקט לא יושלם | קשה | * הקדמת לו"זים משימות * ארגון הזמן בצורה יעילה * להתחיל בדברים היותר קשים | עמידה בזמנים של כתיבת הקוד תוך כדי עמידה בזמנים, בתיק הפרויקט היו קצת עיקובים | 2.5.24 |
| ביצועים גרועים | העלאת והורדת קבצים תיקח זמן רב | קשה | * להיעזר במורה ובחברים מהכיתה כדי למצוא מה מעקב את התוכנית * להריץ כל פעולה בנפרד ולראות האם הבעיה היא כללית או ספציפית | העלאת והורדת קובץ לוקחת זמן קצר | 2.5.24 |
| יציבות | המערכת לא תתפקד עם משתמשים רבים | קשה | * לבדוק האם הבעיה קשורה לכמות המשתמשים או שבכלליות לחיבור של יותר ממשתמש אחד * להריץ פעולות שונות ולראות אם כל הפעולות לא עובדות בחיבור רק של משתמשים או רק פעולות ספציפיות | המערכת מתפקדת היטב גם עם מספר משתמשים רב | 2.5.24 |

# **פרק ג - 'goldrive' - ניתוח**

**1. יכולות בצד לקוח:**

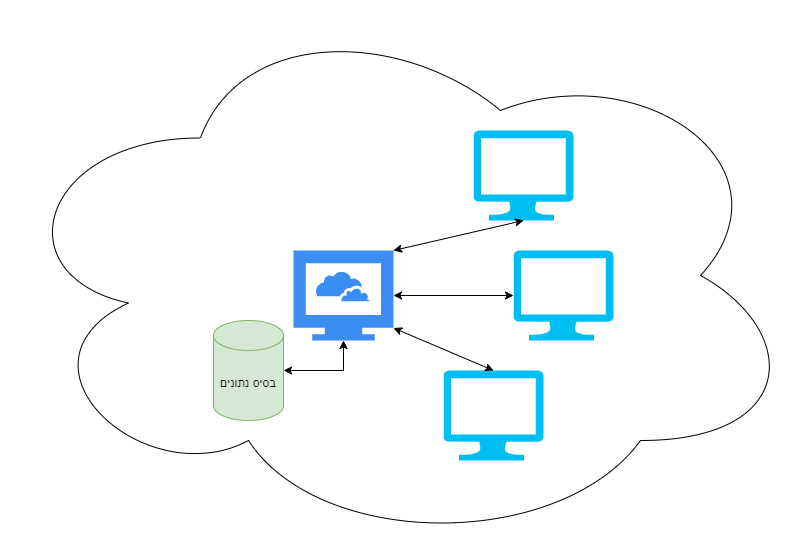
| **שם היכולת** | **מהות היכולת** | **רשימת יכולות** | **רשימת אובייקטים** |
| --- | --- | --- | --- |
| הרשמת משתמש | הרשמת משתמש חדש למערכת | * פתיחת הממשק * קבלת הכנסת פרטי משתמש * הצפנת התוכן * שליחה לשרת * קבלת תשובה מהשרת * פענוח התשובה מהשרת * הצגת התשובה | * תקשורת * הצפנה ופענוח * ממשק משתמש |
| התחברות למשתמש | התחברות למשתמש שקיים במערכת | * פתיחת הממשק * קבלת הכנסת פרטי המשתמש * הצפנת התוכן * שליחה לשרת * קבלת תשובה מהשרת * פענוח התשובה מהשרת * הצגת התשובה | * תקשורת * הצפנה ופענוח * ממשק משתמש |
| העלאת קובץ | העלאת קובץ לענן | * פתיחת ממשק המשתמש * קבלת הקובץ להעלאה * הצפנת הקובץ * שליחה לשרת * קבלת תשובה מהשרת * פענוח התשובה מהשרת * הצגת התשובה | * תקשורת * הצפנה ופענוח * ממשק משתמש * מערכת קבצים |
| הורדת קובץ | הורדת קובץ מהענן | * פתיחת ממשק המשתמש * קבלת הקובץ להורדה * הצפנת שם הקובץ * שליחה לשרת * קבלת תשובה מהשרת * פענוח התשובה מהשרת * הצגת התשובה (הקובץ המורד במידה והצליח) | * תקשורת * הצפנה ופענוח * ממשק משתמש * מערכת קבצים |
| שיתוף קבצים | שיתוף קבצים למשתמשים אחרים | * פתיחת ממשק המשתמש * בחירת קובץ לשיתוף ושם המשתמש אותו רוצה לשתף * הצפנת התוכן * שליחה לשרת * קבלת תשובה מהשרת * פענוח תשובה מהשרת * הצגת התשובה מהשרת | * תקשורת * הצפנה ופענוח * ממשק משתמש * מערכת קבצים |
| מחיקת קובץ | מחיקת קובץ מהענן | * פתיחת ממשק המשתמש * בחירת קובץ למחיקה * הצפנת שם הקובץ * שליחת בקשה למחיקה מהשרת * קבלת תשובה מהשרת * פענוח התשובה מהשרת * הצגת התשובה מהשרת | * תקשורת * הצפנה ופענוח * ממשק משתמש * מערכת קבצים |
| נעילת קבצים | נעילת קבצים באמצעות סיסמא נוספת | * פתיחת ממשק המשתמש * קליטת שם קובץ וסיסמא לנעילה * הצפנת התוכן * שליחה לשרת * קבלת תשובה מהשרת * פענוח תשובה מהשרת * הצגת התשובה מהשרת | * תקשורת * הצפנה ופענוח * ממשק משתמש |
| שכפול קובץ | שכפול קובץ | * פתיחת ממשק המשתמש * קליטת שם קובץ לשכפול * הצפנת שם הקובץ * שליחה לשרת * קבלת תשובה מהשרת * פענוח תשובה מהשרת * הצגת התשובה מהשרת | * תקשורת * הצפנה ופענוח * ממשק משתמש * מערכת קבצים |
| שינוי שם | שינוי שם לקובץ | * פתיחת ממשק המשתמש * קליטת שם הקובץ לשינוי ושם חדש * הצפנת התוכן * שליחה לשרת * קבלת תשובה מהשרת * פענוח תשובה מהשרת * הצגת התשובה מהשרת | * תקשורת * הצפנה ופענוח * ממשק משתמש * מערכת קבצים |

**יכולות בצד שרת:**

| **שם היכולת** | **מהות היכולת** | **רשימת יכולות** | **רשימת אובייקטים** |
| --- | --- | --- | --- |
| הרשמת משתמש | הרשמת משתמש חדש למערכת | * קבלת מידע מהמשתמש * פענוח מידע מהמשתמש * הוספה לבסיס נתונים אם תקין * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים |
| התחברות למשתמש | התחברות למשתמש שקיים במערכת | * קבלת מידע מהמשתמש * פענוח מידע מהמשתמש * בדיקת קיימות בבסיס הנתונים * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים |
| העלאת קובץ | העלאת קובץ לענן | * קבלת מידע מהמשתמש * פענוח מידע מהמשתמש * בדיקת תקינות הקובץ * הוספת הקובץ לענן במידה ותקין * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים * מערכת קבצים |
| הורדת קובץ | הורדת קובץ מהענן | * קבלת מידע מהמשתמש * פענוח המידע מהמשתמש * בדיקת תקינות הקובץ * הצפנת תוכן הקובץ * שליחת תשובה ותוכן הקובץ במידה ותקין | * תקשורת * הצפנה ופענוח * בסיס נתונים * מערכת קבצים |
| שיתוף קבצים | שיתוף קבצים למשתמשים אחרים | * קבלת מידע מהמשתמש * פענוח המידע מהמשתמש * בדיקת תקינות המשתמש לשיתוף בבסיס הנתונים * הוספת הקובץ למשתמש ששותף * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים * מערכת קבצים |
| מחיקת קובץ | מחיקת קובץ מהענן | * קבלת מידע מהמשתמש * פענוח המידע מהמשתמש * בדיקת תקינות הקובץ בענן * מחיקת הקובץ במידה ותקין * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים * מערכת קבצים |
| נעילת קבצים | נעילת קבצים באמצעות סיסמא נוספת | * קבלת מידע מהמשתמש * פענוח המידע מהמשתמש * בדיקת תקינות קובץ וסיסמא * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים |
| שכפול קובץ | שכפול קובץ | * קבלת מידע מהמשתמש * פענוח המידע מהמשתמש * בדיקת תקינות הקובץ * שכפול הקובץ בענן במידה ותקין * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים * מערכת קבצים |
| שינוי שם | שינוי שם לקובץ | * קבלת מידע מהמשתמש * פענוח המידע מהמשתמש * בדיקת תקינות הקובץ והשם החדש * שינוי שם הקובץ במידה ותקין * החזרת תשובה מוצפנת ללקוח | * תקשורת * הצפנה ופענוח * בסיס נתונים * מערכת קבצים |

# **פרק ד - 'goldrive' - עיצוב**

**1 . תיאור החומרה:**

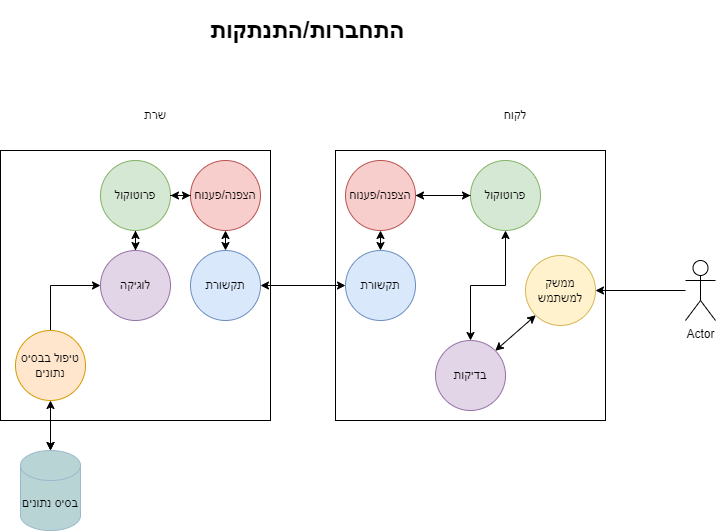
מחשב השרת ומחשבי הלקוחות מבוססי windows ארכיטקטורת התקשורת - שרת מרובה לקוחות, בסיס הנתונים sql מקושר לשרת כל המחשבים נמצאים באותה הרשת.

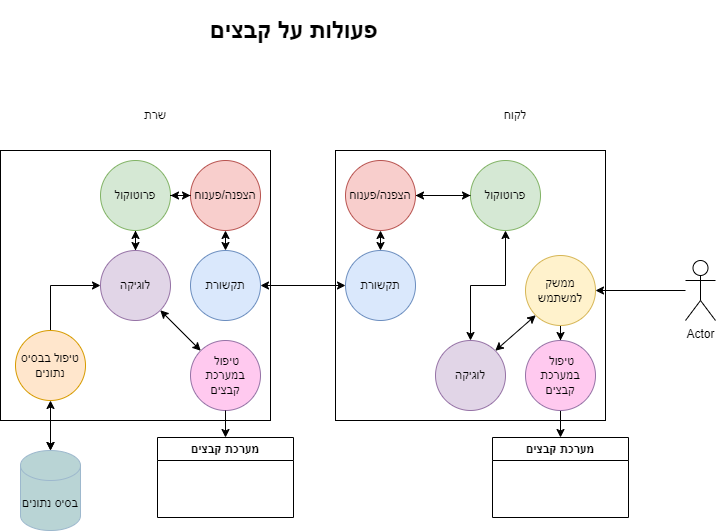
**2. תיאור הטכנולוגיות:**

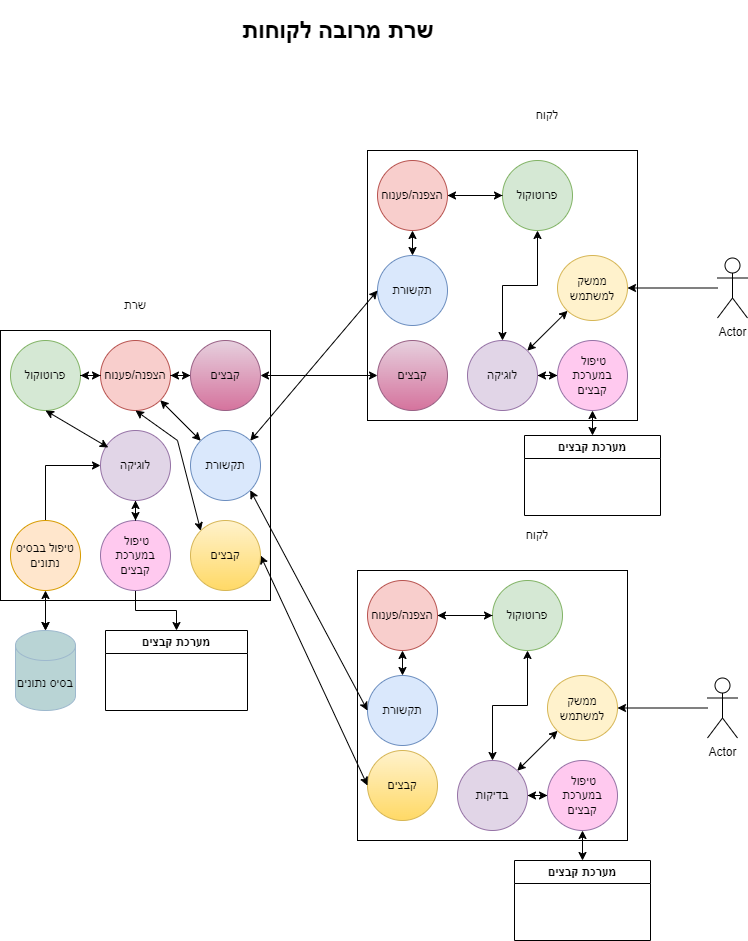
**שפות התכנות-** Python לכתיבת קוד הפרויקט וsql ובעזרת sql אני אממש את בסיס הנתונים.   
**מערכת ההפעלה-** שאני אשתמש בה היא windows.   
תקשורת- שרת מרובה לקוחות מבוסס סוקטים בפרוטוקול TCP.  
**תחומי עניין-**

* הצפנה היברדית (דיפי הלמן ליצירת מפתח סימטרי והצפנה סימטרית)
* שימוש ב thread-ים למימוש עבודה מקבילית בצד שרת והן בצד לקוח
* מערכת קבצים – לטיפול בקבצים המועלים ומורדים מהמערכת

**בסיס נתונים** - SQL בשרת

**3. תיאור תהליכי המערכת**

****

****

**מודלים שאני מייבא:**

| **שם המודל** | **ייעוד** |
| --- | --- |
| **Os** | **טיפול בקבצים** |
| **Sys** | **לסגור את הלקוח באת שגיאה** |
| **Select** | **משמש לטיפול בsocketים בצד שרת** |
| **wxpython** | **גרפיקה** |
| **Threading** | **Threading ו** **multiprocessing** |
| **Queue** | **לתורים שאני אשתמש בהם במהלך הפרויקט** |
| **Sqlite3** | **לטיפול בבסיס הנתונים** |
| **Socket** | **לתקשורת בין הלקוח והשרת** |
| **Pubsub** | **שליחת וקבלת הודעות דרך מערכת בפעלה בין הקבצים** |
| **Secrets** | **על מנת לקבל מספר בצורה רנדומלית** |
| **Shutil** | **מחיקת קבצים** |
| **Time** | **בדיקת זמן ההתחברות לשרת** |
| **Io** | **קריאת תמונות** |
| **Smtplib** | **שליחת מייל באמצעות פייתון** |
| **Zipfile** | **על מנת לארוז תיקיות לקבצי zip** |
| **Base64** | **הצפנה** |
| **Cryptodome** | **הצפנה** |
| **Hashlib** | **הצפנה** |
| **Chardet** | **הצפנה** |
| **Win32file** | **לנתר את הפעולות על קבצים** |
| **Subprocess** | **ליצור סב פרוסס** |
| **Psutil** | **לעבור על הפרוססים הקיימים** |

**מודלים שאני מייצר:**

| **serverProtocol** |
| --- |
| המחלקה שאחראית על הבנייה והפרוק של הפרוטוקול בצד השרת. |
| **תכונות** |
| |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | unpack\_message | msg – ההודעה שהתקבלה מהמשתמש | טאפל שכולל בתוכו את קוד הפרוטוקול ואת המשתנים. | | Get\_data\_from\_string | Msg – ההודעה להוציא ממנה מידע leng – האורך של ההודעה | את המידע שהוצא ואת ההודעה ערוכה | | pack\_files\_message | Username – שם המשתמש אליו שולחים את הקבצים | מחרוזת הקבצים של המשתמש לפי הפרוטוקול. | | pack\_register\_response | Response – התגובה לשליחה username – שם המשתמש password – סיסמת המשתמש email – אימייל המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_login\_response | Response – התגובה לשליחה email – אימייל המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_verify\_ check\_response | Response – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_file\_download\_response | Response – התגובה לשליחה datalen – אורך המידע של הקובץ path – נתיב הקובץ בשרת selected\_path – נתיב הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_upload\_port | port – הפורט להביא למשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_file\_upload\_response | Response – התגובה לשליחה path – נתיב הקובץ להעלות | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_create\_folder\_response | Response – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_delete\_response | Response – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_rename\_file\_response | Response – התגובה לשליחה new\_name – השם החדש של הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_share\_response | Response – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_change\_photo\_response | Response – התגובה לשליחה file\_len – אורך מידע הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_change\_username\_response | Response – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_change\_password\_response | Response – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_change\_email\_response | Response – התגובה לשליחה email – אימייל המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_login\_verify\_response | Status – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_add\_shared\_file | Path – נתיב הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_send\_email\_response | Status – התגובה לשליחה email – אימייל המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_move\_file\_response | Status – התגובה לשליחה new\_folder – מיקום התיקייה אליו הזיזו את הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_paste\_file\_response | Status – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_open\_file\_response | Status – התגובה לשליחה file\_path – נתיב הקובץ data\_len – אורך מידע הקובץ |  | | pack\_get\_details\_response | Status – התגובה לשליחה email – אימייל המשתמש data\_len – אורך מידע הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_delete\_profile\_photo\_response | Status – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_register\_verify\_response | Status – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_check\_code\_response | Status – התגובה לשליחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_forgot\_password\_response | Status – התגובה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_zip\_folder\_response | Status – התגובה לשליחה zipped\_folder\_name – שם הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | |

| **clientProtocol** |
| --- |
| המחלקה שאחראית על הבנייה והפרוק של הפרוטוקול בצד השרת. |
| **תכונות** |
| |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | unpack\_message | msg – ההודעה שהתקבלה מהשרת | טאפל שכולל בתוכו את קוד הפרוטוקול ואת המשתנים. | | get\_data\_from\_string | Line – ההודעה לפירוק length – אורך ההודעה לפירוק | המידע שפורק וההודעה לפירוק לאחר הפירוק. | | Unpack\_files\_message | Msg – ההודעה שהתקבלה מהשרת | מחרוזת הקבצים של המשתמש לפי הפרוטוקול. | | pack\_register\_request | Username – שם משתמש password – סיסמא email – אימייל | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_login\_request | Username – שם משתמש password – סיסמא | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_verify\_check\_request | Email – אימייל המשתמש code – הקוד שהמשתמש הכניס | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_send\_verify\_request | Email – אימייל המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_download\_file\_request | Path – נתיב הקובץ בשרת selected\_path – נתיב הקובץ לשמירה | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_upload\_file\_request | upload\_path – נתיב ההעלאה data\_len – אורך הקובץ | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_create\_folder\_request | upload\_path – נתיב ההעלאה | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_delete\_request | Path – נתיב הקובץ בשרת למחיקה | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_rename\_file\_request | Path – נתיב הקובץ בשרת new\_name – השם לשינוי | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_share\_request | Path – נתיב הקובץ בשרת username – את מי לשתף | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_change\_photo\_request | Username – שם המשתמש file\_len – אורך מידע התמונה | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_change\_password\_request | Username – שם המשתמש old\_password – הסיסמא הישנה new\_password – הסיסמא החדשה confirmed\_password – אימות הסיסמא החדשה | מחרוזת הבנויה לפי הפרוטוקול. | | Pack\_change\_email\_request | Username – שם המתמש Email – אימייל לשינוי code – קוד האימות שנשלח למייל | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_send\_email\_request | Username – שם המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_move\_file\_folder\_request | Src – נתיב הקובץ להזזה dst – נתיב התיקייה להזיז אליה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_paste\_file\_request | Src – נתיב הקובץ להדפסה dst – נתיב התיקייה להדפיס אליה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_open\_file\_request | Path – נתיב הקובץ לפתיחה | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_get\_details\_request | Username – שם המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_delete\_profile\_photo\_request | Username – שם המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_verify\_login\_email\_request | Email – אימייל המשתמש code – קוד האימות שנשלח dont\_ask\_again – האם לבקש אימות שוב במחשב זה username – שם המשתמש | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_verify\_register\_email\_request | Username – שם המשתמש password – הסיסמא email – אימייל המשתמש code – קוד האימות שנשלח dont\_ask\_again – האם לבקש אימות שוב במחשב זה |  | | pack\_check\_code\_request | Email – אימייל המשתמש code – קוד האימות שנשלח במייל | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_forgot\_password\_request | Username – שם המשתמש password – סיסמת המשתמש confirmed\_password – אימות הסיסמא | מחרוזת הבנויה לפי הפרוטוקול. | | pack\_zip\_folder\_request | Folder\_path – נתיב התיקייה לדחוס | מחרוזת הבנויה לפי הפרוטוקול. | |

| **serverComm** |
| --- |
| המחלקה שאחראית על יצירת צד שרת בתקשרות עם הלקוחות |
| **תכונות**   |  |  | | --- | --- | | **שם התכונה** | **שימוש** | | Port | הפורט של השרת | | revcQ | תור ההודעות שהתקבלו לטיפול | | Server\_socket | הסוקט של השרת דרכו עוברת התקשורת | | openClients | מילון הסוקטים המחוברים לשרת עם ערך הip שלהם ומפתח ההצפנה שלהם  open\_clients[client\_socket] =[ip, crypto] | | isRunning | האם השרת רץ | | msgLenBytes | אורך כמות הבתים שאליו מדופנת ההודעה | |
| **פעולות**   |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | \_\_init\_\_ | Port – פורט השרת recvQ – תור ההודעות msgLenBytes – אורך הודעה לשליחה/קבלה | מאתחלת את התכונות ומפעילה את \_main\_loop כ thread | | \_handle\_disconnect | Sock – סוקט לניתוק | מנתק את הלקוח ומוחק אותו מהרשימה open\_clients | | \_find\_socket\_by\_ip | Ip – הכתובת למציאת הסוקט שלו | מחזיר את הסוקט של הלקוח עם הכתובת | | Send | Ip – כתובת הלקוח msg – ההודעה לשליחה | מצפין ושולח את ההודעה ע"פ הפרוטוקול: אורך מדופן ל msgLenBytes ואת ההודעה עצמה | | \_change\_key | client,addr – פרטי הלקוח איתו עושים את ההחלפה | מבצעת החלפת מפתחות עם הלקוח ובסיומה מכניסה את פרטני הלקוח ל open\_clients | | \_close\_server |  | מכבה את isRunning | | \_running\_status |  | מחזיר האם סטטוס השרת | | \_main\_loop |  | מדליק את השרת ומקבל הודעות, מפענח אותן ע"פ מפתח ההצפנה ומכניס tuple ל recvQ הכולל את ה IP של הלקוח ואת ההודעה המפוענחת (ip, msg) | | recv\_file | * **נתיב + שם לשמירה** * **אורך מידע הקובץ (6 בתים)** | תקלוט לתוך הקובץ בנתחים של 1024 בתים את כל הקובץ ותכניס לתור (ip,filenamePath) | | Send\_file | * **Client\_ip – ip המשתמש** * **Params – הנתונים לשליחה** | שולח את הקובץ למשתמש בעזרת הנתונים שהתקבלו | |

| **clientComm** |
| --- |
| **המחלקה שאחראית על יצירת צד לקוח בתקשורת עם השרת** |
| **תכונות**   |  |  | | --- | --- | | **שם התכונה** | **שימוש** | | Port | פורט | | recvQ | תור ההודעות שהתקבלו לטיפול | | my\_socket | הסוקט של הלקוח דרכו עוברת התקשורת | | serverIp | הכתובת של השרת | | Crypt | עצם ההצפנה | | msgLenBytes | אורך כמות הביטים שאליו מדופנת ההודעה | |
| **פעולות**   |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | \_\_init\_\_ | ServerIP – כתובת השרת Port – פורט השרת recvQ – תור ההודעות | מאתחלת את התכונות ומפעילה את \_main\_loop כ thread | | Send | msg – ההודעה לשליחה | מצפין את ההודעה ושולח את ההודעה ע"פ הפרוטוקול: אורך msgLenBytes בתים וההודעה | | \_change\_key |  | מבצעת החלפת מפתחות עם השרת ויוצרת את עצם ההצפנה | | \_close |  | מכבה את isRunning | | \_main\_loop |  | יוצרת את ההתחברות לשרת, מחליפה מפתחות באמצעות \_change\_key ובלולאה מקבלת הודעות מהשרת , מפענחת ומכניסה אותן לתור recvQ | | \_recv\_file | Params – הנתונים שהתקבלו מהשרת | מקבל את מידע הקובץ לפי הפרוטוקול | | Send\_file | Opcode – מספר הפרוטוקול path – נתיב הקובץ currPath – הנתיב הנוכחי בו נמצא המשתמש | שולח קובץ לשרת לפי הפרוטוקול | | Did\_change\_work |  | מחזיר האם מחלפת המפתחות הסתיימה | |

| **serverLogic** | |
| --- | --- |
| **המחלקה שאחראית על הלוגיקה של השרת** | |
| **תכונות** | |
| main\_server | עצם השרת |
| recv\_commands | מילון המכיל עבור כל מפתח (opcode) את שם הפעולה למימוש לדוגמא:  commands['01'] = \_handle\_registeration וכך הלאה |
| myDB | עצם המייצג את בסיס הנתונים של הפרויקט |
| Used\_port | רשימת פורטים שניתנו להעלאת קבצים |
| **פעולות**   |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | main\_loop | יוצרת תור להודעות נכנסות  יוצרת עצם שרת | מפעילה את \_handle\_messages כ thread | | \_handle\_messages | main\_server  recv\_q | יוצרת חיבור לבסיס הנתונים - myDB  מוציאה את ההודעה מתור ההודעות: מפענחת באמצעות הפרוטוקול ומפעילה בהתאם את הפעולה הנדרשת(מפורטת בהמשך) | | \_handle\_registration | username  password  mail  myDB  main\_server  clientIP code\_dic | מנסה להוסיף משתמש לבסיס הנתונים באמצעות הפעולה myDB.add\_user – יוצרת תשובה ע"פ הפרוטוקול ושולחת באמצעות main\_server.send(clientIP, msg) | | \_handle\_email\_register | main\_server db client\_ip code\_dic username password email code dont\_ask\_again |  | | \_handle\_login | Username password myDB main\_server clientIP | בודקת האם הערכים תקינים כדי להתחבר למשתמש – יוצרת תשובה ע"פ פרוטוקול ושולחת אימייל | | \_handle\_email\_login | Main\_server db client\_ip users\_by\_ip code\_dic email code dont\_ask\_again username | בודקת האם קוד האימות שנשלח למייל תקין – יוצרת תשובה ע"פ פרוטוקול ושולחת תשובה | | \_create\_files\_thread | Main\_server db client\_ip users\_by\_ip username | מגרילה פורט שלא נמצא ב used\_ports  בונה הודעה ע"פ הפרוטוקול build\_upload\_port\_request  שולחת באמצעות main\_server.send(ip,msg)  יוצרת תור file\_q מפעילה את recv\_file כ thread | | handle\_files | files\_server files\_q db | שולחת ללקוח את רשימת הקבצים והתיקיות שלו ע"פ הפרוטוקול | | \_handle\_open\_file | path main\_server clientIP | שולחת למשתמש את תוכן הקובץ אותו הוא רוצה לפתוח | | \_handle\_change\_email | Username email code  myDB main\_server clientIP | משנה את המייל בבסיס הנתונים ושולחת תשובה ע"פ הפרוטוקול | | \_handle\_change\_password | Username old\_pass new\_pass confirmed\_pass myDB main\_server clientIP | משנה את הסיסמא בבסיס הנתונים ושולחת תשובה ע"פ הפרוטוקול | | \_handle\_forget\_paddword | Username password confirmed\_password myDB main\_server clientIP | משנה את הסיסמא בבסיס הנתונים ושולחת תשובה ע"פ הפרוטוקול | | \_handle\_rename\_file | Path newName db main\_server clientIP | משנה את שם הקובץ בדרייב ומעדכנת את הלקוח ע"פ הפרוטוקול | | \_handle\_share\_file | username  Path users\_by\_ip main\_server clientIP db | משתפת את הקובץ עם המשתמש שהתקבל ומעדכנת את הלקוח ע"פ הפרוטוקול | | \_handle\_delete\_file | db Path main\_server clientIP | מוחקת את הקובץ המבוקש ומעדכנת את הלקוח ע"פ הפרוטוקול | | \_handle\_download\_file | Selected\_path path main\_server clientIP | שולחת ללקוח את הקובץ ע"פ הפרוטוקול | | \_handle\_upload\_file | path data main\_server clientIP | יוצרת את הקובץ אצל השרת ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_create\_dir | db path main\_server clientIP | יוצרת תיקייה אצל השרת ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_move\_file | Main\_server db client\_ip src dst | מעבירה קובץ אצלי קבצי השרת ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_paste\_file | Main\_server db client\_ip src dst | מוסיפה את הקובץ שהודבק לקבצי הלקוח ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_get\_details | Main\_server client\_ip db username | שולחת ללקוח את פרטי המשתמש ותוכן התמונת פרופיל | | \_send\_email | Main\_server my\_db client\_ip code\_dic email | מגרילה מספר בין 6 ספרות ושולח למייל של המשתמש שהתקבל | | \_handle\_change\_photo | Files\_server client\_ip username photo\_data | משנה את תמונת הפרופיל של המשתמש אצל השרת ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_delete\_profile\_photo | Files\_server client\_ip username | מוחקת את תמונת הפרופיל של המשתמש אצל השרת ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_send\_email | Main\_server db client\_ip code\_dic username | מפעילה את שליחת המייל ושולחת תגובה ללקוח ע"פ הפרוטוקול | | \_handle\_check\_email | Main\_server db client\_ip code\_dic email code | בודקת האם הקוד שהוכנס זהה לקוד שנשלח למייל של הלקוח ומחזירה תגובה ע"פ הפרוטוקול | | \_handle\_zip\_folder | Main\_server db client\_ip folder\_path | אורזת את התיקייה שנבחרה ומחזירה תגובה ללקוח ע"פ הפרוטוקול | | |

|  |
| --- |
| **clientLogic** |
| **המחלקה שאחראית על הלוגיקה של הלקוח** |
| **פעולות**   |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | Main\_loop |  | יוצרת את תור ההודעות ובונה את עצם הלקוח | | \_Handle\_messages | msg\_q | מוציאה את ההודעה מתור ההודעות ומפעילה את הפעולה הנכונה בהתאם | | \_handle\_files | Msg\_q | מוציאה את ההודעה מתור ההודעות ומפעילה את הפעולה הנכונה בהתאם | | \_show\_register\_dialog | Status username password email | מראה תגובה בגרפיקה בהתאם על פי התשובה | | \_handle\_register | status | משנה את הגרפיקה על פי התשובה | | \_show\_login\_dialog | Status email | מראה תגובה בגרפיקה בהתאם על פי התשובה | | \_handle\_login | status | משנה את הגרפיקה על פי התשובה | | \_handle\_change\_email | Status email | משנה את הגרפיקה על פי התשובה | | \_handle\_change\_password | status | משנה את הגרפיקה על פי התשובה | | \_handle\_forget\_paddword | status | משנה את הגרפיקה על פי התשובה | | \_handle\_rename\_file | Status new\_name | משנה את הגרפיקה על פי התשובה | | \_handle\_share\_file | status | משנה את הגרפיקה על פי התשובה | | \_handle\_delete\_file | status | משנה את הגרפיקה על פי התשובה | | \_handle\_download\_file | Status path selected\_path data | משנה את הגרפיקה על פי התשובה | | \_handle\_open\_file | File\_comm status server\_path data | פותחת את הקובץ לעריכה או לצפייה בהתאם לסוג הקובץ | | \_handle\_upload\_file | Status path | משנה את הגרפיקה על פי התשובה | | \_handle\_create\_dir | status | משנה את הגרפיקה על פי התשובה | | Handle\_files\_port\_ | Port | פותחת thread חדש לטיפול בקבצים | | \_handle\_add\_shared\_file | Path | מוסיף את הקובץ לקבציי הלקוח | | \_handle\_move\_file | Status path | משנה את הגרפיקה על פי התשובה | | \_handle\_paste\_file | Status | משנה את הגרפיקה על פי התשובה | | \_handle\_get\_files | Email photo | משנה את הערכים אצל המשתמש | | \_handle\_change\_photo | Data | משנה את ערך תמונת הפרופיל אצל המשתמש | | \_handle\_username\_check | Status email | מראה את דיאלוג האימות אם תקין או שגיאה | | \_handle\_code\_check | Status | מראה את דיאלוג שינוי הסיסמא או שגיאה | | \_handle\_forgot\_password | Status | מראה תגובה ללקוח האם היה תקין או לא | | \_handle\_zip\_file | Status zip\_name | משנה את הגרפיקה ע"פ התשובה | | \_handle\_send\_files | Branches | משנה את קצבי הלקוח | |

| **sFileHandler** |
| --- |
| **המחלקה שאחראית על פעולות הקבצים מצד השרת** |
| **פעולות**   | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | --- | --- | --- | | Rename\_file | Path – נתיב הקובץ newname – שם חדש | משנה את שם הקובץ | | Delete\_file | Path – נתיב הקובץ | מוחק את הקובץ | | Generate\_unique\_zip \_name | Path – נתיב הקובץ base\_name – שם הקובץ | בודק את שמות קבצי הזיפ שיש בתיקייה ולפי זה יוצר את שם הזיפ הבא | | create\_zip | Total\_path – נתיב הקובץ | יוצר את קובץ הזיפ מחזיר 1 אם לא עבד 0 אם עבד | |

| **DB** |
| --- |
| **המחלקה שאחראית על טיפול במבנה הנתונים** |
| **תכונות**   |  |  | | --- | --- | | **שם התכונה** | **שימוש** | | Conn | מצביע לבסיס הנתונים | | Cursor | המצביע בתוך בסיס הנתונים על מנת לעשות פעולות | | DBname | שם בסיס הנתונים |   **פעולות**   | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | --- | --- | --- | | \_\_init\_\_ |  | יוצרת את עצם בסיס הנתונים וקוראת ל \_create\_table | | \_create\_data\_base |  | מתחברת לבסיס הנתונים ויוצרת את טבלת המשתמשים users אם לא קיימת | | username\_exist | Username – שם המשתמש | בודק האם שם המשתמש קיים בטבלה מחזיר 0/1 לפי אמת או שקר | | add\_user | Username – שם המשתמש password – סיסמא email - מייל | מוסיף משתמש לטבלה אם הוא לא קיים באמצעות \_username\_exists ומחזיר תשובה 1/0 | | Delete\_user | Username – שם משתמש | מסיר משתמש מהטבלה אם הוא קיים ומחזיר תשובה 1/0 | | Change\_password | Username – שם המשתמש password – סיסמא | משנה את הסיסמא של המשתמש אם הוא קיים ומחזיר תשובה 1/0 | | Change\_username | Username – שם המשתמש newname – השם החדש | משנה את שם המשתמש אם הוא קיים  ומחזיר תשובה 1/0 | | Change\_email | Username – שם המשתמש email – אימייל | משנה את האימייל של המשתמש אם הוא קיים  ומחזיר תשובה 1/0 | | Get\_password | Username – שם המשתמש | מחזיר את הסיסמא של המשתמש אם הוא קיים | | Get\_email | Username – שם המשתמש | מחזיר את המייל של המשתמש אם הוא קיים | | get\_ips\_of\_user | Username – שם המשתמש | מחזיר את הכתובות בהם המשתמש היה מחובר מהם | | add\_remembered\_ip | Username – שם המשתמש ip – כתובת המשתמש | מוסיף את הכתובת למשתמש בטבלת הכתובות של המשתמשים | |

| **encryption** |
| --- |
| **המחלקה שאחראית על תהליכי ההצפנה והפענוח בפרויקט** |
| **תכונות**   |  |  | | --- | --- | | **שם התכונה** | **שימוש** | | Key | ממפתח ההצפנה | | Bs | גודל הבלוק | |
| **פעולות**   |  |  |  | | --- | --- | --- | | **שם הפעולה** | **טענת כניסה** | **טענת יציאה** | | \_\_init\_\_ | Key – מפתח ההצפנה | יוצר את האובייקט | | \_pad | message – ההודעה | ריפוד ההודעה | | \_unpad | message – ההודעה | הורדת הריפוד | | Enc\_msg | message – הודעה להצפנה | ההודעה המוצפנת | | Dec\_msg | Encrypt\_Msg – הודעה לפענוח | ההודעה הרגילה | | Hash\_msg | Msg – הודעה לגיבוב | **פעולה חיצונית** – ערך hash של ההודעה | | Get\_DH\_factor |  | **פעולה חיצונית** -מחזירה מספר ראשוני b לטובת החלפת מפתח בשיטת דיפי-הלמן | | Create\_symetry\_key | Private\_key – מפתח ההצפנה הפרטי shared\_key – מפתח ההצפנה המשותף | **פעולה חיצונית** - מייצרת מפתח סימטרי – מחזירה עצם encryption | |

| **PortsHandler** | |
| --- | --- |
| **המחלקה אשר אחראית על יצירת הפורטים שמשמשים להעלאת והורדת קבצים באמצעות התקשרות** | |
| **תכונות** | |
| **שם התכונה** | **שימוש** |
| **Unused\_ports** | **גנרטור שמחזיר את הפורט הבא שאינו בשימוש בשרת** |
| **פעולות** | |
| **שם הפעולה** | **טענת יציאה** |
| **Get\_next\_port** | **מחזירה את הפורט הבא שאינו בשימוש באמצעות unused\_ports** |

| **monitorFile** | | |
| --- | --- | --- |
| **המחלקה אשר אחראית על יצירת הפורטים שמשמשים להעלאת והורדת קבצים באמצעות התקשרות** | | |
| **פעולות** | | |
| **שם הפעולה** | **טענת כניסה** | **טענת יציאה** |
| **Monitor** | **Path\_to\_watch – נתיב התיקייה אשר אנחנו רוצים לבדוק בה שינויים q – התור בו אנחנו שמים את העדכונים** | **שמה בq האם היה שינוי בקובץ בתיקייה שבה אנחנו מסתכלים עליה** |
| **Get\_all\_pid** | **Process\_name – שם הprocess שאנחנו רוצים לקבל את כל הpids שלו** | **מחזיר את כל הpids עם שם הprocess** |
| **Wait\_until** | **File\_path – נתיב הקובץ q – התור בו אנחנו שמים את העדכונים** | **פותח את הקובץ וכאשר הקובץ נסגר שם בq שהתוכנה נסגרה** |

|  |  |
| --- | --- |
| **ClassSettings** | |
| **המחלקה שבה יש את התכונות המשמשות להרצת הקוד בכיתה** | |
| **תכונות** | |
| **שם התכונה** | **שימוש** |
| **SERVERPORT** | **הפורט להתחברות לשרת בכיתה** |
| **SERVERIP** | **Ip השרת להתחברות בכיתה** |
| **USER\_FILES\_PATH** | **נתיב הקבצים להרצת הגרפיקה של הלקוח בכיתה** |
| **SERVER\_FILES\_PATH** | **נתיב קבצי המשתמשים אצל השרת בכיתה** |
| **USER\_PROFILE\_PHOTOS** | **נתיב תמונות הפרופיל של המשתמשים אצל השרת בכיתה** |

|  |  |
| --- | --- |
| **HomeSettings** | |
| **המחלקה שבה יש את התכונות המשמשות להרצת הקוד בבית** | |
| **תכונות** | |
| **שם התכונה** | **שימוש** |
| **SERVERPORT** | **הפורט להתחברות לשרת בבית** |
| **SERVERIP** | **Ip השרת להתחברות בבית** |
| **USER\_FILES\_PATH** | **נתיב הקבצים להרצת הגרפיקה של הלקוח בבית** |
| **SERVER\_FILES\_PATH** | **נתיב קבצי המשתמשים אצל השרת בבית** |
| **USER\_PROFILE\_PHOTOS** | **נתיב תמונות הפרופיל של המשתמשים אצל השרת בבית** |

|  |
| --- |
| **CurrentSettings** |
| **המחלקה משתמשת בתכונות של HomeSettings ושל ClassSettings ועוזרת לי להחליף בין ההגדרות בבית לבין ההגדרות בכיתה באמצעות שינוי של שורה אחת ולא של כל מופע של התכונות בקוד** |

|  |
| --- |
| **FileMenuFeatures** |
| **תפריט האופציות לפעולות על הקבצים של המשתמש (למחוק קובץ, לשנות שם, לשתף קובץ וכו')** |

|  |
| --- |
| **FolderMenuFeatures** |
| **תפריט האופציות לפעולות על התיקיות של המשתמש (לפתוח את התייקיה, למחוק את התיקייה וכו')** |

|  |
| --- |
| **UserMenuFeatures** |
| **תפריט האופציות לפעולות של המשתמש (העלאת קובץ, יצירת תיקייה וכו')** |

|  |
| --- |
| **ProfileSettingsMenu** |
| **תפריט האופציות להגדרות המשתמש (שינוי תמונת פרופיל ומחיקת תמונת פרופיל)** |

|  |
| --- |
| **ChangePasswordDialog** |
| **דיאלוג אישי לשינוי הסיסמא** |

|  |
| --- |
| **ConfirmMailDialog** |
| **דיאלוג אישי לאימות המייל בעזרת הקוד שנשלח למייל** |

|  |
| --- |
| **ForgotPasswordDialog** |
| **דיאלוג אישי לשכחתי את הסיסמא** |

|  |
| --- |
| **TransparentText** |
| **טקסט ללא רקע** |

|  |
| --- |
| **MyDropTarget** |
| **דרופ טארגט שבעזרתו בודקים גרירה של קבצים** |

|  |
| --- |
| **FilesPanel** |
| **מסך הגרפיקה של קבצי המשתמש שיש בו את כל קבצי ותיקיות המשתמשדד** |

|  |
| --- |
| **LoginPanel** |
| **מסך הגרפיקה של הכנסת פרטי המשתמש על מנת להתחבר למשתמש** |

|  |
| --- |
| **MainFrame** |
| **הפריים הראשי של מסכי הגרפיקה של המשתמש** |

|  |
| --- |
| **MainPanel** |
| **הפאנל הראשי של מסכי הגרפיקה של המשתמש עליו שמים את שאר המסכים** |

|  |
| --- |
| **RegistrationPanel** |
| **מסך הגרפיקה של הכנסת פרטי המשתמש על מנת ליצור למשתמש** |

|  |
| --- |
| **UserPanel** |
| **מסך הגרפיקה שבו המשתמש יכול לראות את פרטי המשתמש שלו, ולשנות שם את תמונת הפרופיל שלו** |

**4. תיאור סביבת הפיתוח:**

**שפות התכנות-** שפות התכנות שאני אשתמש בהם הם **פיתון וsql**, אני אעשה שימוש בsql על מנת לעבוד מול בסיס נתונים של הפרויקט.  
ובפיתון בשביל קבצי הקוד של הפרויקט.  **כלי הפיתוח והכלים** **לבדיקה**- כלי הפיתוח והכלים לבדיקה שאני אשתמש בהם בפרויקט הם **PyCharm**, **Wireshark**, **DB Browser for SQLite** וב**file explorer**. אני אשתמש בPyCharm על מנת לערוך את קבצי הקוד שלי, בWireshark על מנת לבדוק ולעבוד על התקשורת וההצפנה שלי, בDB Browser for SQLite בשביל לבדוק את בסיס הנתונים שלי ובfile explorer על מנת לראות ולבדוק את שינויי הקבצים.

**5. סוגיות מרכזיות בפרויקט:**

**סוגיה** – מימוש התקשורת בפרויקט

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | סוקט יחיד דרכו כולם מחוברים לשרת | קל למימוש | לא יעיל, מונע טיפול מקבילי בלקוחות/בקשות שונות |
| 2 | סוקט לתקשורת כללית וסוקט לקבצים | הפרדה ברורה בין קבצים לשאר הדברים ולא נדרש למיין זאת  העלאת/הורדת קבצים לא מעכבת התחברות לקוחות אחרים | קצת יותר מורכב למימוש  ברגע שכמה לקוחות רוצים בו זמנית לעלות/להוריד קבצים נוצר עיכוב |
| 3 | סוקט לתקשורת כללית ולכל לקוח סוקט קבצים משלו | הפרדה מוחלטת בין הלקוחות ולכן מונע עיכובים בין משתמשים  הפרדה ברורה בין קבצים לשאר הדברים ולא נדרש למיין זאת | יותר סוקטים יותר מורכב לניהול |

החלופה שבחרתי בה היא חלופה 3 – בפרויקט שלי, העיסוק העיקרי הוא עם קבצים ולכן חשוב שלא יהיו עיכובים/ תקיעות / ערבוב עם מידע אחר.

**סוגיה** - סוג ההצפנה בפרויקט

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | הצפנה סימטרית | מהירה, קלה ליישום | דורש הסכמה על מפתח ההצפנה שצריך לדעת מראש |
| 2 | הצפנה אסימטרית | נוחה לשימוש ובטוחה יותר | איטית יחסית בעיקר בעבודה עם קבצים |
| 3 | הצפנה היברידית (יצירת המפתח הסימטרי בפרוטוקול דיפי הלמן) | פותרת את בעיית העברת המפתח הסימטרי ומהירה | קשה יותר ליישום |

החלופה שבחרתי היא חלופה 3 – בפרויקט שלי, העיסוק הוא עם קבצים והצפנה א-סימטרית יכולה לגרום לעיכובים ולבעיות איטיות, אני אשתמש בפרוטוקול דיפי-הלמן ליצירת מפתח סימטרי וכך אוכל להשתמש בהצפנה הסימטרית שהיא מהירה ונוחה.

**סוגיה** - העברת המידע על הקבצים

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | העברת המידע על כל הקבצים של המשתמש בבת אחת | חוסך בתקשורת משום שיש אצל הלקוח את כל המידע | מידע גדול מאוד ועלול לגרום לבעיות של איבוד מידע העברת מידע שברובו עלול להיות מיותר |
| 2 | העברת רק השמות בהתחלה ואחר כך שליחת המידע על הקובץ המתבקש | נוח משום שמעבירה רק את המידע החיוני באמת ויותר קלה לטיפול | יותר תקשורת |

החלופה שבחרתי היא חלופה 2 – משום שאני משתמש בקבצים העברת כל המידע תהיה כבדה מאוד ותעמיס על הרשת ואילו העברת רק הקבצים הנחוצים יותר קלה לטיפול ויותר יעילה

**סוגיה** - אופן שמירת הקבצים **של המשתמשים בשרת**

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | כל הקבצים של כל המשתמשים נמצאים במחיצה אחת בצד שרת | קל לשימוש | נדרשת יכולת בצד השרת לדעת אלו קבצים שייכים לאיזה משתמש, פעולה שיכולה לקחת זמן בכל פנייה של משתמש כלשהו לקבצים שלו – יחייב פיתוח אלגוריתם כלשהו בצד שרת |
| 2 | עבור כל משתמש בצד שרת, תהייה מחיצה עם שם המשתמש (שהוא חד ערכי במערכת) שם ישמרו הקבצים שלו | יוצרת סדר וארגון טובים יותר - גישה בצד שרת לקבצי משתמש מהירה ולא מצריכה אלגוריתם כלשהו | בזבוז מקום בצד שרת  במקרים של שינוי שם משתמש במערכת אצרך לשנות גם את שם התיקייה שלו |
| 3 | שמירת הקבצים בבסיס הנתונים | הקבצים הרבה יותר בטוחים, גישה קלה יותר אליהם | מנפחת את בסיס הנתונים ולכן ברוב בסיסי הנתונים גם אצלי שמירת קבצים לא אפשרית בבסיס הנתונים |

החלופה שבחרתי בה היא חלופה 2 –בחלופה זו יש סדר וארגון בצד שרת שמאיצים את הטיפול בבקשות משתמש לקבצים שלו שזה מהות הפרויקט אצלי.

**סוגיה** – אופן שמירת הסיסמא בבסיס הנתונים

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | הצפנה | נוח לשימוש משום שמפתח ההצפנה ידוע | איפה שומרים את מפתח ההצפנה ובמידה ומגלים את המפתח מתגלות כל הסיסמאות שבמערכת |
| 2 | פונקציית גיבוב | קלה לבדיקה ובטוחה מאוד | יכולה לגרום להתנגשויות והכנסת סיסמא לא נכונה בכל זאת תכניס את המשתמש למערכת |
| 3 | כמו שהוכנס על ידי המשתמש | קלה מאוד לקריאה ובדיקה | לא בטוחה בעליל |

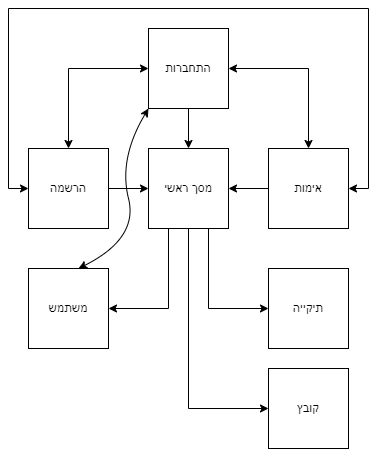
החלופה שבחרתי בה היא חלופה 2 – משום שהסיכויים להתנגשות מאוד נמוכים ובמידה והתקיימה התנגשות אחת היא יותר בטוחה מאשר גילוי מפתח ההצפנה או החוסר בהצפנה.

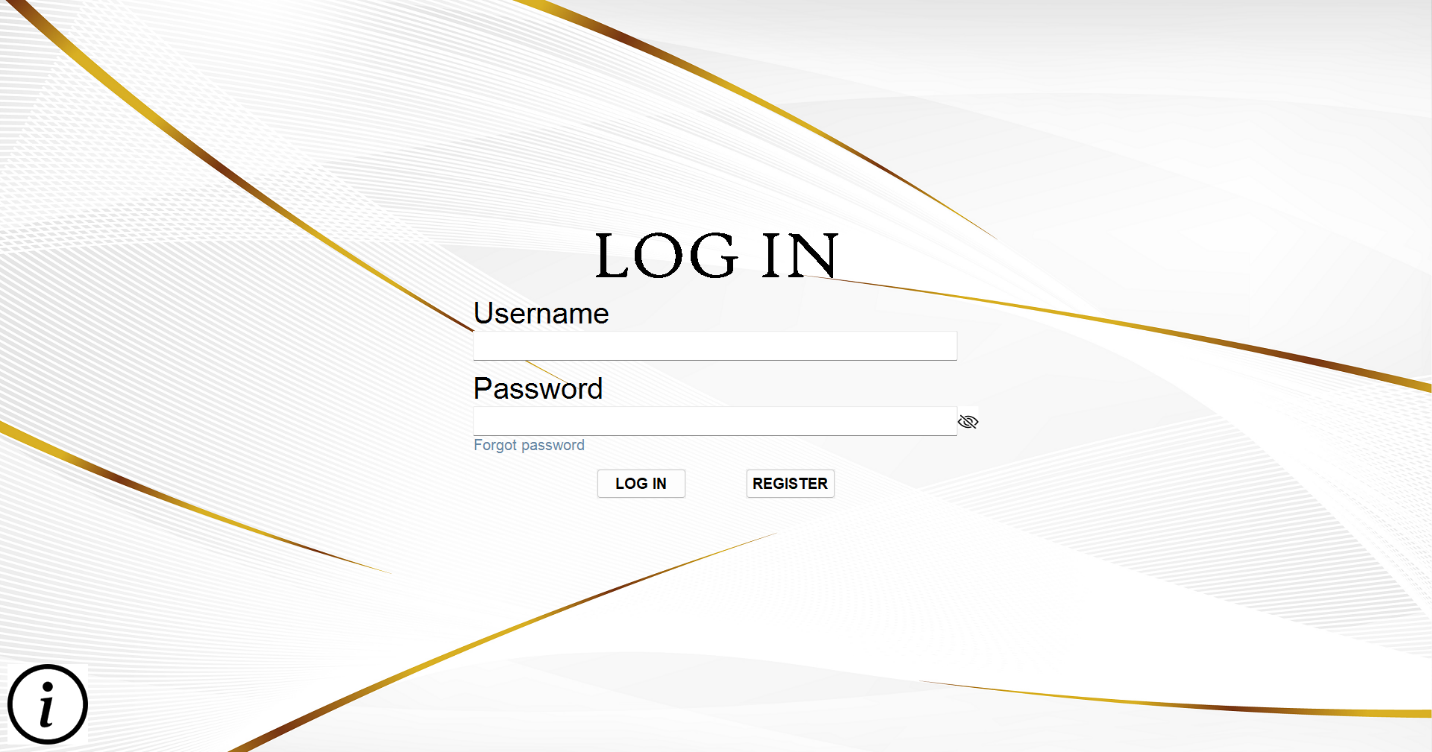
**סוגיה** – אופן העברת תיאור מבנה הקבצים של המשתמש ללקוח (לטובת הצגתם בממשק המשתמש)

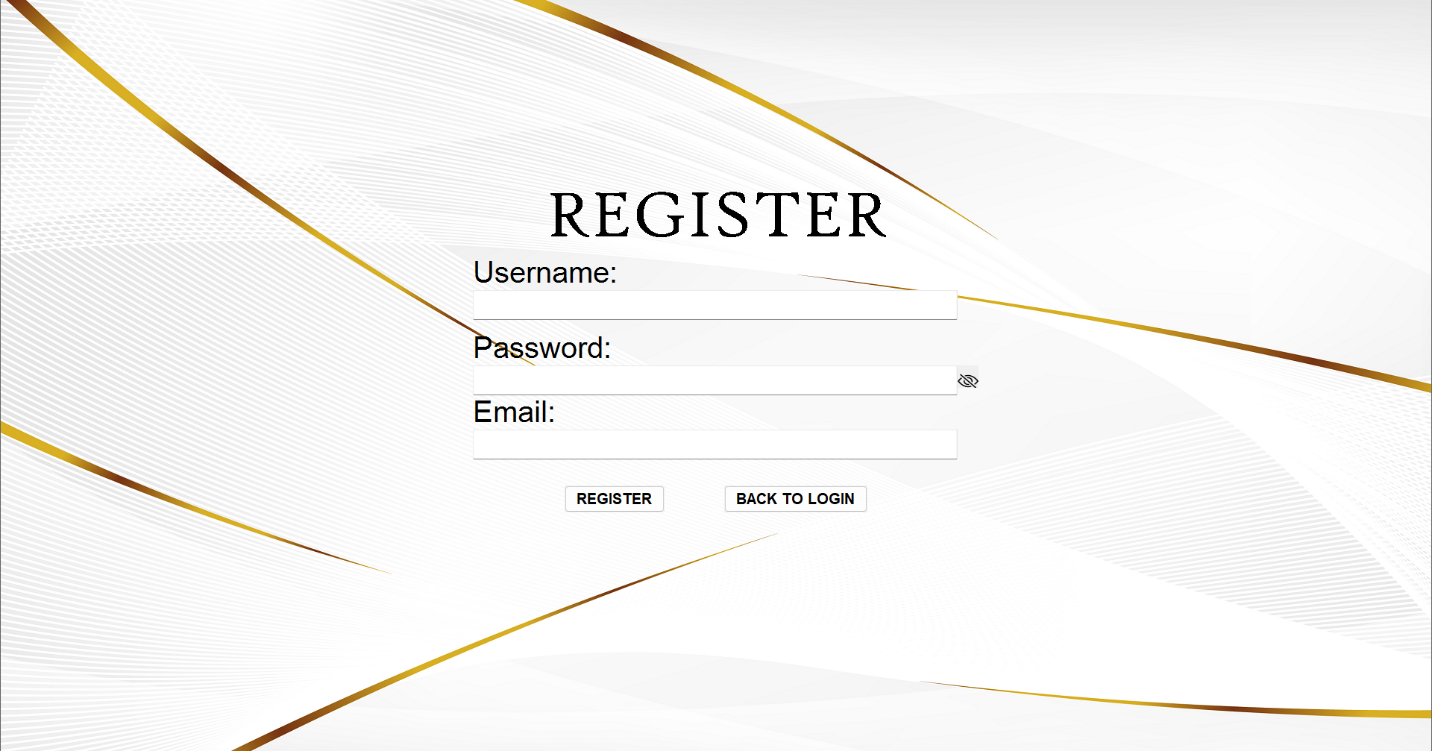
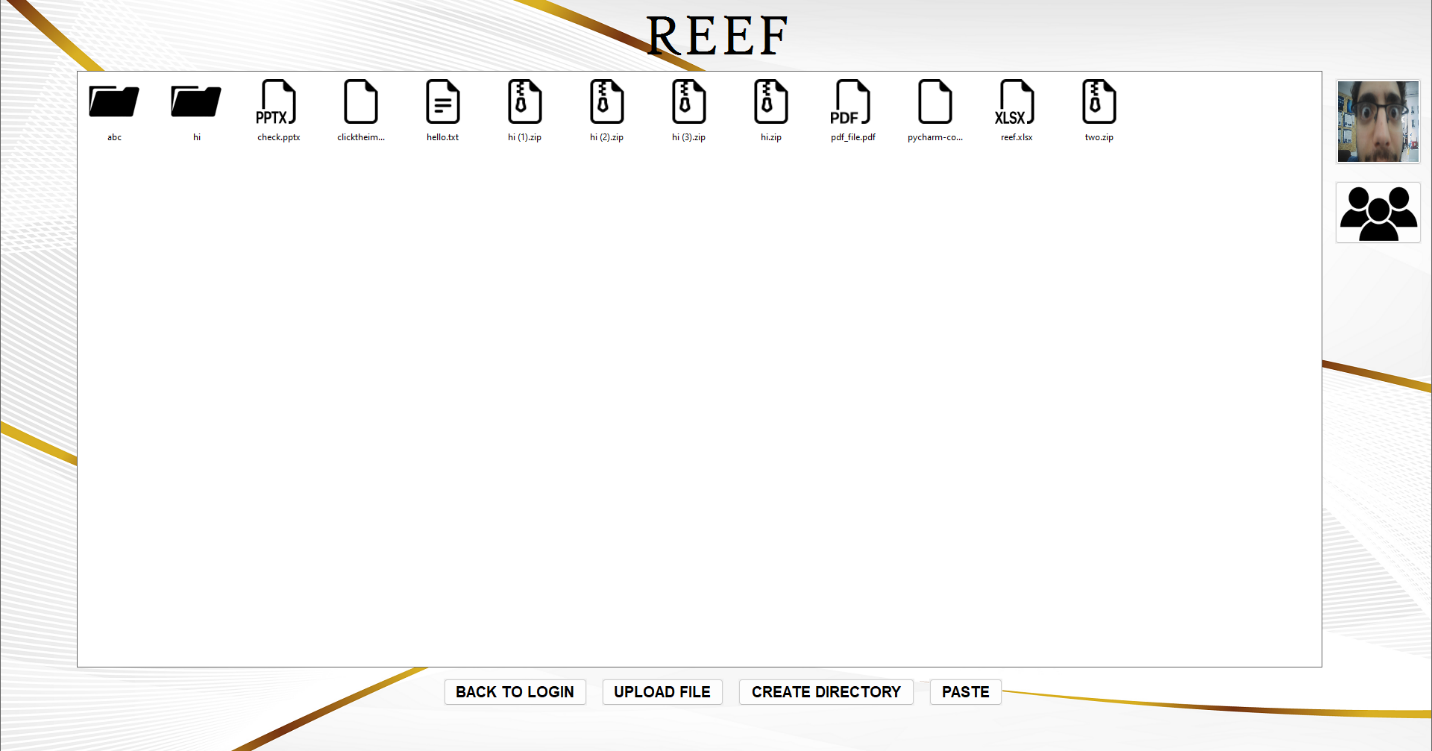
| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | העברת רק התיקייה הנוכחית (עליה כרגע המשתמש מסתכל) | קל למימוש  מיקום הלקוח בתיקיות נמצא אצל השרת | השרת מנהל את תצוגת הקבצים של הלקוח – עומס על השרת  ישפיע על תגובות התצוגה |
| 2 | העברת מבנה הקבצים הכולל ללקוח | נוחה ומאפשרת רק שליחה אחת  כל ניהול המעבר בין התיקיות בצד לקוח לא מעמיס על השרת | נדרש לממש פרוטוקול לייצוג מבנה הקבצים של המשתמש  בכל שליחה של פעילות במערכת הקבצים תידרש העברת ה מסלול כולו |

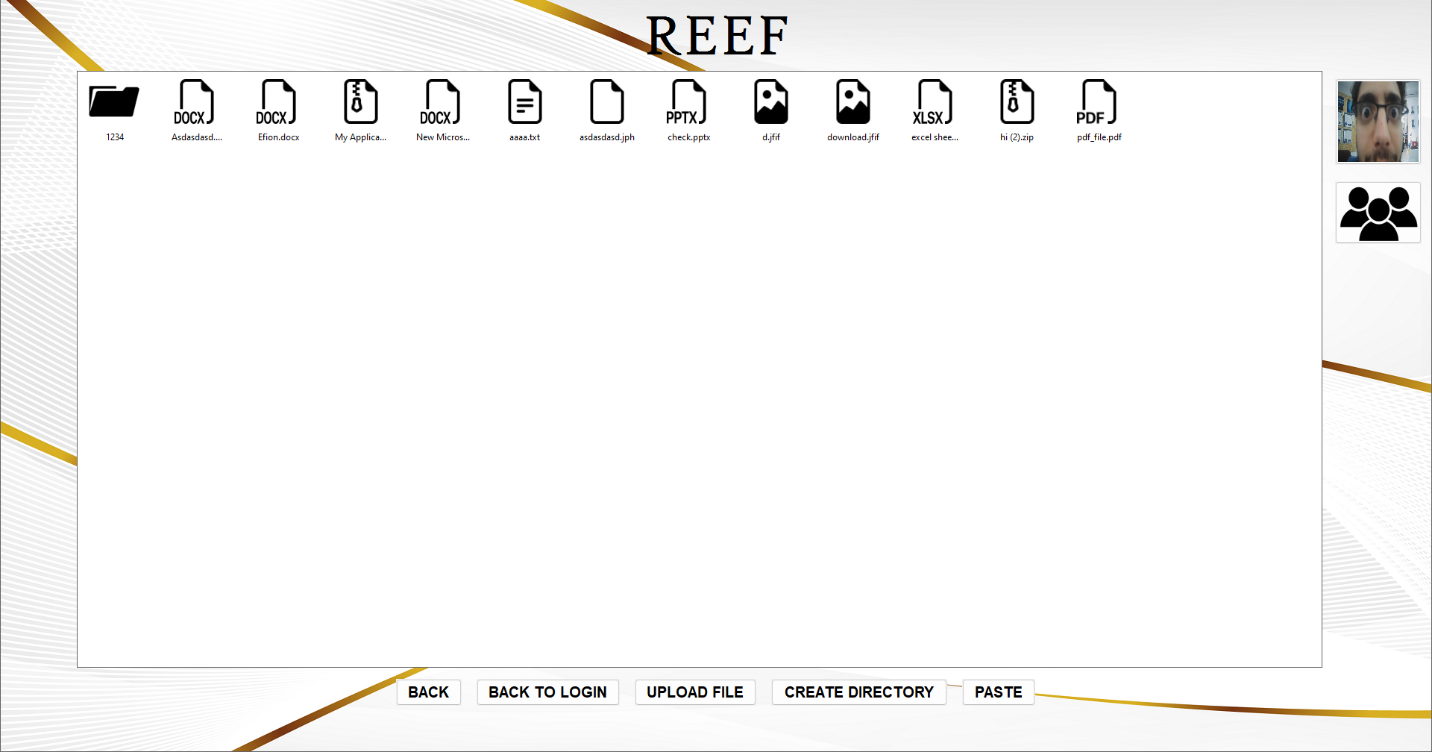
החלופה שבחרתי בה היא חלופה 2 – משום שבפרויקט שלי חשובה מהירות התגובה ואם אני שולח הכל בהתחלה זה מאפשר לתוכנת הלקוח להגיב במהירות לבקשות המשתמש זה הרבה יותר מהיר ומוריד תקשורת ולחץ מהשרת

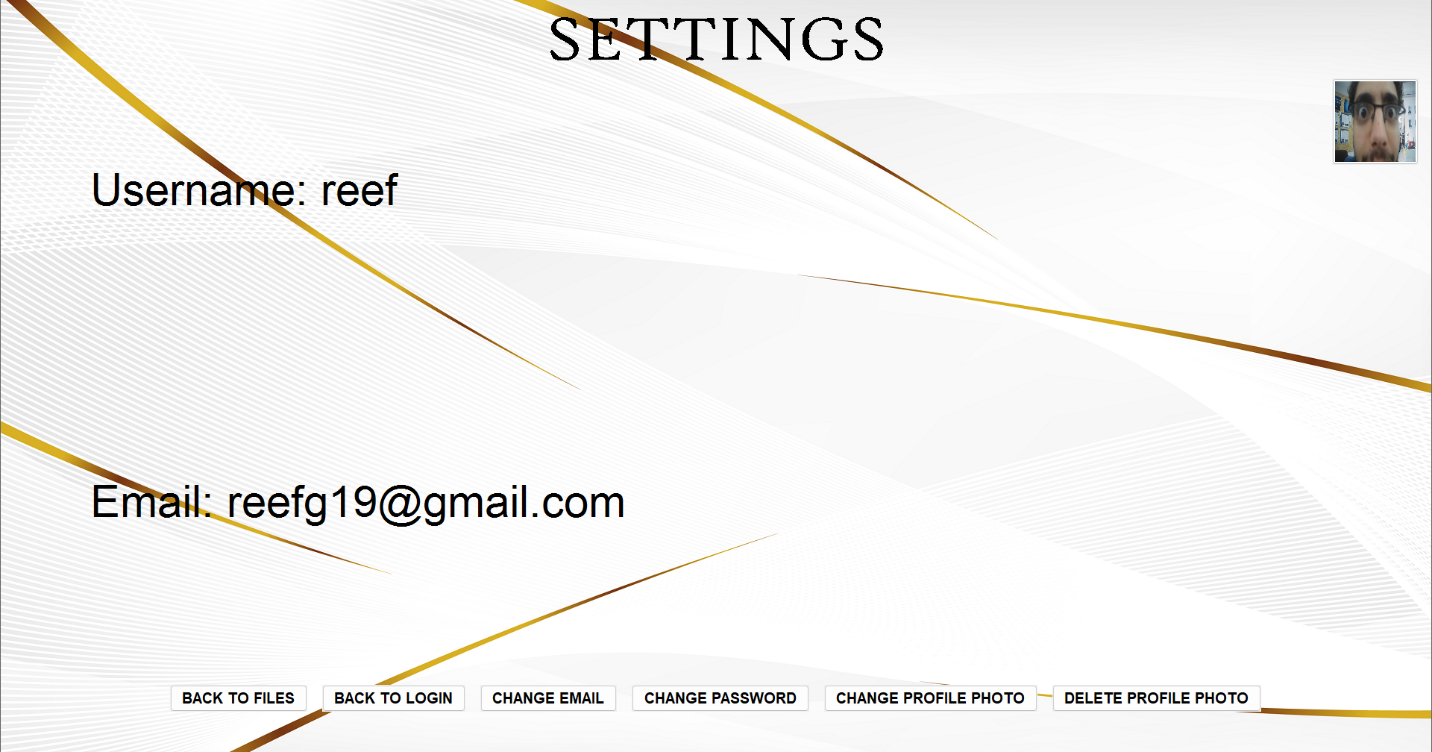
**6. תרשים מסכי הפרויקט**

א. תרשים עץ המסכים:  


**מסך התחברות**: ****

**מסך יצירת משתמש:מסך הקבצים הראשי:** 

**מסך של תיקייה: **

**מסך פרטי המשתמש: **

**7. פרוטוקול תקשורת**

**בסוקט הודעות כלליות** שליחת הודעה:  
תמיד: אורך הודעה מוצפנת – 2 בתים, הודעה עצמה מוצפנת

**בסוקט הקבצים** שליחת הודעה:  
תמיד: אורך הודעה מוצפנת – 10 בתים, הודעה עצמה מוצפנת

| **שם הפרוטוקול** | **שולח** | **נמען** | **opcode** | **שדות ההודעה** | **דוגמא** |
| --- | --- | --- | --- | --- | --- |
| **הרשמה** | **לקוח** | **שרת** | **01** | * **אורך שם משתמש (2 בתים)** * **שם המשתמש** * **אורך סיסמא (2 בתים)** * **סיסמא** * **אורך מייל (2 בתים)** * **מייל** | **0104reef06reef1114 reef@gmail.com** |
| **הרשמה** | **שרת** | **לקוח** | **01** | **0/1 הצליח/נכשל** | **010** |
| **התחברות** | **לקוח** | **שרת** | **02** | * **אורך שם משתמש (2 בתים)** * **שם המשתמש** * **אורך סיסמא (2 בתים)** * **סיסמא** | **0204reef07reef981** |
| **התחברות** | **שרת** | **לקוח** | **02** | **0/1 הצליח/נכשל** | **021** |
| **שליחת שמות הקבצים במחיצה** | **שרת** | **לקוח** | **03** | * **אורך שם התיקייה (2 בתים)** * **שם התיקייה** * **אורך שמות התיקיות שבתיקייה (4 בתים)** * **שמות התיקיות שבתיקייה מופרד על ידי !** * **אורך שמות הקבצים שבתיקייה (4 בתים)** * **שמות הקצבים שבתיקייה מופרד על ידי @** | **01b0003a!c0019file2.txt@03 hello.txt** |
| **שינוי תמונת פרופיל** | **לקוח** | **שרת** | **04** | * **אורך שם משתמש חדש (2 בתים)** * **שם משתמש חדש** | **0408Reef6213** |
| **שינוי תמונת פרופיל** | **שרת** | **לקוח** | **04** | **0/1 הצליח/נכשל** | **040** |
| **שינוי אימייל** | **לקוח** | **שרת** | **05** | * **אורך אימייל חדש (2 בתים)** * **אימייל חדש** | **0518Reef6213@gmail.com** |
| **שינוי אימייל** | **שרת** | **לקוח** | **05** | **0/1 הצליח/נכשל** | **051** |
| **שינוי סיסמא** | **לקוח** | **שרת** | **06** | * **אורך סיסמא חדשה (2 בתים)** * **סיסמא חדשה** | **0608Reef@41!** |
| **שינוי סיסמא** | **שרת** | **לקוח** | **06** | **0/1 הצליח/נכשל** | **060** |
| **שליחת אימייל לאימות** | **לקוח** | **שרת** | **07** | * **אורך המייל (2 בתים)** * **מייל** | **0717reefg19@gmail.com** |
| **שליחת אימייל לאימות** | **שרת** | **לקוח** | **07** | **0/1 הודעה נשלחה כן/לא** | **071** |
| **שינוי שם קובץ** | **לקוח** | **שרת** | **08** | * **אורך שם ישן + הנתיב (2 בתים)** * **שם ישן + הנתיב** * **אורך שם חדש (2 בתים)** * **שם חדש** | **0816folder/text1.txt12fold er12.txt** |
| **שנוי שם קובץ** | **שרת** | **לקוח** | **08** | **0/1 הצליח/נכשל** | **081** |
| **שיתוף קובץ** | **לקוח** | **שרת** | **09** | * **אורך שם + נתיב (2 בתים)** * **שם + נתיב** * **אורך שם אליו משתפים (2 בתים)** * **שם אליו משתפים** | **0915reef/myText.txt06 friend** |
| **שיתוף קובץ** | **שרת** | **לקוח** | **09** | **0/1 הצליח/נכשל** | **090** |
| **מחיקת קובץ** | **לקוח** | **שרת** | **10** | * **אורך שם + נתיב (2 בתים)** * **שם + נתיב** | **1016files/text1.docx** |
| **מחיקת קובץ** | **שרת** | **לקוח** | **10** | **0/1 הצליח/נכשל** | **100** |
| **הורדת קובץ** | **לקוח** | **שרת** | **11** | * **אורך שם + נתיב (2 בתים)** * **שם + נתיב** | **1108text.txt** |
| **הורדת קובץ** | **שרת** | **לקוח** | **11** | * **אורך מידע הקובץ (6 בתים)** * **מידע הקובץ** | **110000131001011011101** |
| **העלאת קובץ** | **לקוח** | **שרת** | **12** | * **אורך נתיב + שם לשמירה (2 בתים)** * **נתיב + שם לשמירה** * **אורך מידע הקובץ (6 בתים)** * **מידע הקובץ** | **1216random/file1.txt000012 101101011010** |
| **העלאת קובץ** | **שרת** | **לקוח** | **12** | **0/1/2 הצליח/קובץ כבר קיים/נכשל** | **121** |
| **יצירת מחיצה** | **לקוח** | **שרת** | **13** | * **אורך נתיב + שם לשמירה (2 בתים)** * **נתיב + שם לשמירה** | **1314folders/reefs** |
| **יצירת מחיצה** | **שרת** | **לקוח** | **13** | **0/1/2 הצליח/שם תיקייה קיים/נכשל** | **130** |
| **עידכון הוספת קובץ** | **שרת** | **לקוח** | **14** | * **אורך נתיב + שם (2 בתים)** * **נתיב + שם** | **1410reefs/page** |
| **אימות במייל** | **לקוח** | **שרת** | **15** | * **אורך קוד (2 בתים)** * **קוד** | **1513reefg19@gmail0512345** |
| **אימות במייל** | **שרת** | **לקוח** | **15** | **/01 הצליח/נכשל** | **150** |
| **פורט להעלאת קובץ** | **שרת** | **לקוח** | **16** | **מספר פורט** | **161200** |
| **בקשה לאימות** | **לקוח** | **שרת** | **17** | * **אורך שם משתמש (2 בתים)**   **שם משתמש** | **1704reef** |
| **בקשה לאימות** | **שרת** | **לקוח** | **17** | * **אורך סטטוס ההצלחה (2 בתים)** * **0/1 הצליח או נכשל** * **אורך המייל לשליחה (2 בתים)** * **המייל לשליחה** | **1701117reefg19@gmail.com** |
| **הזזת קובץ** | **לקוח** | **שרת** | **18** | * **אורך נתיב המקור (2 בתים)** * **נתיב המקור** * **אורך נתיב להזזה (2 בתים)** * **נתיב להזזה** | **1806origin03dst** |
| **הזזת קובץ** | **שרת** | **לקוח** | **18** | * **אורך קוד התשובה (2 בתים)** * **0/1 נכשל או הצליח** * **אורך נתיב הקובץ שהוזז אליו (2 בתים)** * **הקובץ שהוזז אליו** | **1801003dst** |
| **הדבקת הקובץ** | **לקוח** | **שרת** | **19** | * **אורך נתיב המקור (2 בתים)** * **נתיב המקור** * **אורך נתיב להזזה (2 בתים)** * **נתיב להזזה** | **1910originfile04dest** |
| **הדבקת קובץ** | **שרת** | **לקוח** | **19** | **0/1 הצליח או נכשל** | **191** |
| **פתיחת קובץ** | **לקוח** | **שרת** | **20** | * **אורך נתיב הקובץ לפתיחה (2 בתים)** * **נתיב הקובץ לפתיחה** | **2009fld/file1** |
| **פתיחת קובץ** | **שרת** | **לקוח** | **20** | * **אורך התגובה (2 בתים)** * **0/1 הצליח/נכשל** * **אורך נתיב הקובץ (2 בתים)** * **נתיב הקובץ** * **אורך אורך מידע הקובץ (2 בתים)** * **אורך מידע הקובץ** | **2001006folder041982** |
| **קבלת פרטים** | **לקוח** | **שרת** | **21** | * **אורך שם משתמש (2 בתים** * **שם משתמש** | **2104reef** |
| **קבלת פרטים** | **שרת** | **לקוח** | **21** | * **אורך התגובה (2 בתים)** * **1/0 הצליח/נכשל** * **אורך מייל (2 בתים)** * **מייל** * **אורך אורך תמונת הפרופיל (2 בתים** * **אורך תמונת הפרופיל** | **21010017reefg19@gmail.co m042512** |
| **מחיקת תמונת פרופיל** | **לקוח** | **שרת** | **22** | * **אורך שם משתמש (2 בתים)** * **שם משתמש** | **2204reef** |
| **מחיקת תמונת פרופיל** | **שרת** | **לקוח** | **22** | **1/0 הצליח/נכשל** | **220** |
| **אימות התחברות במייל** | **לקוח** | **שרת** | **23** | * **אורך מייל (2 בתים)** * **2 בתים** * **אורך קוד אימות (2 בתים)** * **קוד אימות** * **אורך האם לשאול במחשב זה שוב (2 בתים)** * **האם לשאול שוב במחשב זה** * **אורך שם משתמש (2 בתים)** * **שם משתמש** | **2317reefg19@gmail.com 0641234104True04reef** |
| **אימות הרשמה במייל** | **לקוח** | **שרת** | **24** | * **אורך שם משתמש (2 בתים)** * **שם משתמש** * **אורך סיסמא (2 בתים)** * **סיסמא** * **אורך מייל (2 בתים)** * **2 בתים** * **אורך קוד אימות (2 בתים)** * **קוד אימות** * **אורך האם לשאול במחשב זה שוב (2 בתים)** * **האם לשאול שוב במחשב** | **2404reef051234517reefg1 9@gmail.com0661231204 True** |
| **אימות הרשמה במייל** | **שרת** | **לקוח** | **24** | **0/1 הצליח/נכשל** | **240** |
| **בדיקת קוד אימות** | **לקוח** | **שרת** | **25** | * **אורך מייל (2 בתים)** * **מייל אורך קוד (2 בתים)** * **קוד** | **2513abc@gmail.com061241 61** |
| **בדיקת קוד אימות** | **שרת** | **לקוח** | **25** | **0/1 הצליח/נכשל** | **251** |
| **שינוי סיסמא באמצעות שכחתי סיסמא** | **לקוח** | **שרת** | **26** | * **אורך שם משתמש (2 בתים)** * **שם משתמש** * **אורך סיסמא (2 בתים)** * **סיסמא** * **אורך אימות סיסמא (2 בתים)** * **אימות סיסמא** | **2604reef08reef124508reef1 245** |
| **שינוי סיסמא באמצעות שכחתי סיסמא** | **שרת** | **לקוח** | **26** | **0/1 הצליח/נכשל** | **260** |
| **דחיסת תיקייה לקובץ** | **לקוח** | **שרת** | **27** | * **אורך נתיב הקובץ (2 בתים)** * **נתיב הקובץ** | **2712reef/folder1** |
| **דחיסת תיקייה לקובץ** | **שרת** | **לקוח** | **27** | * **אורך התגובה (2 בתים)** * **תגובה** * **אורך שם הקובץ הדחוס (2 בתים)** * **שם הקובץ הדחוס** | **2701012reef (1).zip** |
| **הודעה על התנתקות** | **שרת** | **לקוח** | **28** |  | **28** |

**8. תיאור מבני הנתונים של הפרויקט:**

* **Queue – ישמשו להעברת ההודעות הנכנסות מהתקשורת ללוגיקה**
* **Dictionary – ישמשו לאחזקת פרטי הלקוחות המחוברים, פעולות לטיפול בהודעות נכנסות ועוד.**
* **List – ישמשו להחזקת פרטי משתמשים**
* **Tuple – המידע שיוכנס לתורים יכלול תמיד גם מידע מזהה על הלקוחות לדוגמא (ip, msg)**

מאגר המידע שאשתמש בו הוא מערכת הקבצים של windows. **במחשב השרת** לכל משתמש תהיה תיקייה בשם המשתמש שלו שתכיל את כל הקבצים והתיקיות שלו.

מסד הנתונים שאשתמש בו הוא מסד נתונים מסוג SQL בעל טבלה אחת

שם מסד הנתונים: **GolDrive**

הטבלה: טבלת המשתמשים

שם הטבלה: users  
מפתח הטבלה: שם המשתמש username  
שדת הטבלה:

* **username – שם המשתמש מטיפוס String**
* **password – סיסמת המשתמש מוצפנת באמצעות hash מטיפוס String**
* **email – מייל המשתמש מטיפוס String**

**דוגמא:**

|  |  |  |
| --- | --- | --- |
| **email** | **password** | **username** |
| **עד 50 תווים** | **עד 10 תווים** | **עד 10 תווים** |
| **reefg19@gmail.com** | **av123ffas151** | **1Reef** |
| **Imrih2006@gmail.com** | **ta12mam61** | **Imri24** |

**9. היבטי אבטחה בפרויקט – חולשות והדרך לטפל בהן**

| **שם החולשה** | **תיאור החולשה** | **פתרון** |
| --- | --- | --- |
| **Sql injection** | בעזרת הכנסות קלט מסוימות המשתמש עלול להשתמש בהם על מנת לקבל פרטים חשאיים ולחדור למערכת | במימוש שאילתות ה SQL אדאג לשימוש בקוד שלא יאפשר הכנסת מידע מזיד לשאילתא |
| **MITM** | מישהו עלול להאזין לתקשורת ולגלות פרטיים חשאיים | הצפנת כלל המידע שעובר ברשת ובנוסף מתקבלת הודעת אישור על כל בקשה מהשרת או מהלקוח מהצד השני |
| **DOS/DDOS** | הצפת השרת בהודעות התחברות | המערכת לא תאפשר הרבה ניסיונות של התחברות מכתובת אחת בפרק זמן קצר ולאחר מספיק ניסיונות תחסום את הכתובת |
| **התחזות** | משתמש עלול לגלות פרטים של משתמש ולפרוץ לחשבון שלו | התוכנה תשתמש באימות דו שלבי ותשלח מייל למשתמש אליו ניסו להתחבר ורק לאחר הכנסת הקוד הסודי שנשלח למייל יוכל להתחבר לחשבון |

# **פרק ה - 'goldrive' – הקוד**

**1 .**מימוש הסוקטים בפרויקט

**class** PortsHandler:  
 unused\_ports = (port **for** port **in** range(1235, 2235))  
  
 @staticmethod  
 **def** get\_next\_port():  
 *"""* **:return***: returns the next port in the generator  
 """* **return** next(PortsHandler.unused\_ports)

סוג ההצפנה בפרויקט

**import** base64  
**from** Cryptodome.Cipher **import** AES  
**from** Cryptodome **import** Random  
**import** hashlib  
**import** random  
**import** chardet  
  
  
**class** Encryption(object):  
 **def** \_\_init\_\_(self, key):  
 *"""* **:param** *key: the encryption key  
 """* self.bs = AES.block\_size  
 self.key = hashlib.sha256(str(key).encode()).digest()  
  
 **def** enc\_msg(self, message):  
 *"""* **:param** *message: the message to encrypt* **:return***: the encrypted message  
 """* **if** type(message) == str:  
 message = message.encode()  
 raw = self.\_pad(message)  
 iv = Random.new().read(AES.block\_size)  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
 **return** base64.b64encode(iv + cipher.encrypt(raw))  
  
 **def** dec\_msg(self, encrypt\_message):  
 *"""* **:param** *encrypt\_message: the encrypted message to decrypt* **:return***: the decrypted message  
 """* enc = base64.b64decode(encrypt\_message)  
 iv = enc[:AES.block\_size]  
 cipher = AES.new(self.key, AES.MODE\_CBC,iv)  
 a = self.\_unpad(cipher.decrypt(enc[AES.block\_size:]))  
 encoding = chardet.detect(a)  
 **if** encoding[**'encoding'**] == **"ascii"**:  
 a = a.decode()  
 **return** a  
  
 **def** \_pad(self, message):  
 *"""* **:param** *message: message* **:return***: the message with the padding  
 """* **return** message + (self.bs - len(message) % self.bs) \* chr(self.bs - len(message) % self.bs).encode()  
  
 @staticmethod  
 **def** \_unpad(message):  
 *"""* **:param** *message: encrypted message* **:return***: the encrypted message without the padding  
 """* **return** message[:-ord(message[len(message) - 1:])]  
  
  
**def** hash\_msg(msg):  
 *"""* **:param** *msg: the msg to hash* **:return***: the hash of the msg  
 """* **return** hashlib.sha3\_256(msg.encode()).hexdigest()  
  
  
**def** get\_dh\_factor():  
 *"""* **:return***: creates the dh factor  
 """* private\_key = random.randint(1, p)  
  
 **return** private\_key, (g \*\* private\_key) % p  
  
  
**def** create\_symmetry\_key(private\_key, shared\_key):  
 *"""* **:param** *private\_key: the personal key* **:param** *shared\_key: the key i got from the server* **:return***: returns the symmetrical encryption object from the dh key change  
 """* **return** Encryption((shared\_key \*\* private\_key) % p)  
  
  
p = 7723  
g = 1229

העברת המידע על הקבצים

**def** pack\_files\_message(username):  
 *"""* **:param** *username: username of user* **:return***: string with all files and directories of user in the drive  
 """* files\_of\_user = **"03"**path = **f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"  
  
 for** (dirname, dirs, files) **in** os.walk(path):  
 *# removing the path so that it's only the directories in the system and not from the server* dirname = dirname[len(path) + 1::].replace(**'\\'**, **'/'**)  
  
 *# adding the length of the name and the name* files\_of\_user += str(len(dirname)).zfill(2) + dirname  
  
 *# adding the length of the directories and the directories list* dirs = **'!'**.join(dirs)  
 files\_of\_user += str(len(dirs)).zfill(4) + dirs  
  
 *# adding the length of the files and the files list* files = **'@'**.join(files)  
 files\_of\_user += str(len(files)).zfill(4) + files  
  
 files\_of\_user += **'\n'  
  
 return** files\_of\_user

אופן שמירת הקבצים **של המשתמשים בשרת**

**def** \_handle\_email\_register(main\_server, db, client\_ip, code\_dic, username, password, email, code, dont\_ask\_again):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary containing the code sent to an email* **:param** *username: the username of the client* **:param** *password: the password of the client* **:param** *email: the email of the client* **:param** *code: the code entered* **:param** *dont\_ask\_again: does the user want to verify his email in his computer* **:return***: try's adding the user to the database and sends response to the client  
 """* ans = 2  
  
 **if** email **in** code\_dic:  
 **if** code\_dic[email] == code:  
 **if** 4 <= len(username) <= 10 **and** len(password) >= 4:  
 ans = db.add\_user(username, password, email)  
  
 **if** ans == 0 **and** dont\_ask\_again == **"True"**:  
 db.add\_remembered\_ip(username, client\_ip)  
  
 msg = serverProtocol.pack\_register\_verify\_response(ans)  
 main\_server.send(client\_ip, msg)  
  
 **if** ans == 0:  
 **if not** os.path.isdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"**):  
 os.mkdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"**)  
 os.mkdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}/@#$SHAREDFILES$#@"**)

אופן שמירת הסיסמא של המשתמש בבסיס הנתונים

**def** add\_user(self, username, password, email):  
 *"""* **:param** *username: username* **:param** *password: password* **:param** *email: email* **:return***: adds user to table  
 """* flag = 1  
 **if not** self.username\_exist(username):  
 sql = **"INSERT INTO users VALUES (?,?,?)"** self.curr.execute(sql, (username, encryption.hash\_msg(password), email))  
 self.conn.commit()  
  
 flag = 0  
 **return** flag

אופן העברת תיאור מבנה הקבצים של המשתמש ללקוח (לטובת הצגתם בממשק המשתמש)

**def** pack\_files\_message(username):  
 *"""* **:param** *username: username of user* **:return***: string with all files and directories of user in the drive  
 """* files\_of\_user = **"03"** path = **f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"  
  
 for** (dirname, dirs, files) **in** os.walk(path):  
 *# removing the path so that it's only the directories in the system and not from the server* dirname = dirname[len(path) + 1::].replace(**'\\'**, **'/'**)  
  
 *# adding the length of the name and the name* files\_of\_user += str(len(dirname)).zfill(2) + dirname  
  
 *# adding the length of the directories and the directories list* dirs = **'!'**.join(dirs)  
 files\_of\_user += str(len(dirs)).zfill(4) + dirs  
  
 *# adding the length of the files and the files list* files = **'@'**.join(files)  
 files\_of\_user += str(len(files)).zfill(4) + files  
  
 files\_of\_user += **'\n'  
  
 return** files\_of\_user

בדיקה מתי הלקוח סוגר את הקובץ שהוא רצה לפתוח מתוך האפליקציה

def get\_all\_pid(process\_name):  
 current = []  
 for p in psutil.process\_iter():  
 if p.name() == process\_name:  
 current.append(p.pid)  
  
 return current  
  
  
def wait\_until(file\_path, q):  
 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'  
  
 ls1 = get\_all\_pid(process\_name)  
 subprocess.Popen(['start', file\_path], shell=True)  
  
 while True:  
 ls2 = get\_all\_pid(process\_name)  
 if ls2 != ls1:  
 break  
  
 new\_pid = set(ls2) - set(ls1)  
 pid = list(new\_pid)[0]  
  
 while psutil.pid\_exists(pid):  
 pass  
  
 q.put("Finished")

# **פרק ו - 'goldrive' – בדיקות('קופסא לבנה')**

| **מספר** | **שם הבדיקה** | **מטרת הבדיקה** | **מה נרדש לבצע** | **מתי** | **מה בוצע בפועל** |
| --- | --- | --- | --- | --- | --- |
| **1** | **התחברות** | **האם אפשר להתחבר למשתמש כמו שצריך** | **לפתוח את המערכת, להתחבר עם שם משתמש וסיסמא נכונים ולראות שמצליחים להיכנס למערכת.**  **לחזור על הפעולה עם שם משתמש וסיסמא שגויה ולראות שלא מצליחים להיכנס למערכת** | **7.5.24** | **הכניסה למשתמש מתאפשרת רק בהכנסת פרטי משתמש תקינים** |
| 2 | העלאת קבצים | האם אפשר לשמור קבצים למשתמש | לפתוח את המערכת ולהתחבר עם שם סיסמא ומשתמש, לנסות להעלות קובץ ולראות שהוא עלה ומוצג בקבצים שלי, במידה והוא עלה להתנתק מהמשתמש ולהתחבר שוב ולראות שהקובץ עדיין מוצג. | 7.5.24 | העלאת קבצים מתאפשרת ונשמרת גם להתחברות הבאה |
| 3 | הורדת קבצים | האם אפשר להוריד קבצים | לפתוח את התוכנה, להתחבר למשתמש, לבחור קובץ ולנסות להוריד אותו, לבדוק האם הקובץ מורד כמו שצריך ובמקום הנכון. | 7.5.24 | הקובץ מורד להכין שהמשתמש בחר ומכיל את התוכן |
| 4 | איש שלישי | האם אפשר לקרוא מידע שעובר | לפתוח Wireshark ואז לפתוח את התוכנה, להתחבר למשתמש ולנסות להעלות קובץ אחר כך לבדוק Wireshark האם אפשר לקרוא את המידע שעבר או שהוא מוצפן. | 12.5.24 | המידע שעובר בתקשורת מוצפן היטב ואינו ניתן לקריאה. |
| 5 | שיתוף קבצים | האם אפשר לשתף קובץ עם חבר | לפתוח את התוכנה להתחבר למשתמש, לבחור קובץ ולשתף אותו עם משתמש אחר, להתחבר למשתמש האחר ולראות האם הקובץ ששותף מופיע באזור השיתופים. | 7.5.24 | ניתן לשתף קבצים עם משתמשים אחרים. מופיע בזמן אמת. |
| 6 | מחיקת קבצים | האם אפשר למחוק קבצים | לפתוח את התוכנה, להתחבר למשתמש, לבחור קובץ ולנסות למחוק אותו, לראות שהקובץ נמחק ולא מופיע, להתחבר מחדש למשתמש ולראות שהקובץ עדיין לא מופיע. | 7.5.24 | ניתן למחוק קבצים ולא מופיע גם לאחר התנתקות והתחברות מחדש. |
| 7 | שינוי שם | האם אפשר לשנות שם לקובץ | לפתוח את התוכנה, להתחבר למשתמש, לבחור קובץ ולנסות לשנות לו את השם, לראות שהשם משתנה כמו שצריך, להתחבר מחדש ולראות שהשם נותר משונה. | 7.5.24 | ניתן לשנות לקובץ השינוי מופיע גם לאחר התנתקות והתחברות מחדש. |
| 8 | כניסה כפולה | לבדוק האם המערכת מונעת ממשתמש אחד להיכנס בו זמנית ממחשבים שונים | לפתוח את התוכנה, ליצור משתמש, לראות שמתחבר כמו שצריך, ולנסות להתחבר לאותו משתמש ממחשב אחר ולראות שזה **לא** מתחבר. | 12.5.24 | אי אפשר להתחבר מ2 מחשבים לאותו משתמש באותו הזמן. |
| 9 | בדיקת ביצועים | לבדוק האם המערכת מבצעת פעולות במהירות | לפתוח את התוכנה, להתחבר למשתמש ולנסות לבדוק כל פעולה שמתאפשרת לקובץ ולבדוק שהכל נעשה במהירות. | 7.5.24 | כל הפעולות נעשות בצורה מהירה. |
| 10 | בדיקת יציבות | לבדוק האם המערכת מתפקדת בקביעות למרות שיש מספר משתמשים מחוברים | לפתוח את התוכנה במספר מחשבים, להתחבר מכל מחשב למשתמש ומכל משתמש להריץ איזושהי פעולה על קובץ, לראות שבכל משתמש הפעולה נעשתה כמו שצריך. | 7.5.24 | המערכת מתפקדת בקביעות גם כשיש מספר משתמשים מחוברים |
| 11 | זמני תגובה | לבדוק האם יש הבדלים בזמני התגובה של המערכת כאשר מורידים/מעלים קבצים בגדלים שונים | לפתוח את התוכנה, להתחבר למשתמש ולנסות להעלות ולהוריד קבצים שונים בעלי גודל שונה ולבדוק את הזמן שלוקח. | 7.5.24 | יש הבדל בזמני התגובה רק כאשר מדובר בקובץ גדול מאוד, אחרת העבודה עם הקבצים היא מהירה מאוד. |

# **פרק ז - 'goldrive' – מדריך למשתמש**

**A screenshot of a login screen

Description automatically generatedכאשר המשתמש יפתח את התוכנה ייפתח לו מסך ההרשמה שבו יוכל המשתמש להתחבר למשתמש שקיים או ללחוץ על כפתור ההרשמה על מנת לעבור למסך בו ניתן ליצור משתמש חדש.**

**A screenshot of a register

Description automatically generated**

כאשר המשתמש יוצר משתמש חדש עליו לכתוב את כתובת המייל איתו הוא מעוניין להירשם, יישלח אל כתובת המייל שלו קוד אימות שהוא יצטרך להכניס בתוכנה, כאשר הוא יכניס את קוד האימות התוכנה תישאל איתו האם הוא רוצה שבמחשב זה בכל פעם שהוא מתחבר התוכנה תבקש ממנו אימות במייל או שרק חד פעמי. במידה ויבחר שלא, בכל פעם שיתחבר תינתן האפשרות לעשות זאת.

כאשר המשתמש מתחבר לתוכנה בפעם הראשונה ייפתח לו מסך הקבצים הראשי שלו, במסך זה הוא יכול להעלות קבצים ליצור תיקיות להדביק קבצים אחרי שמעתיקים אותם ולחזור למסך ההתחברות.

הפעולות שהמשתמש יכול לבצע על קבצים באמצעות לחיצה ימנית על הקובץ הם:

1. למחוק
2. לשנות שם
3. להוריד למחשב
4. לשתף עם משתמשים אחרים
5. להעתיק
6. לערוך/לפתוח (תלוי לפי סוג הקובץ)

ניתן גם להזיז קובץ לתיקייה אחרת על ידי גרירתו באמצעות העכבר על ידי לחיצה שמאלית על הקובץ שמתבקש מבלי לשחרר ולשחרר מעל התיקייה אליו רוצים להעביר את הקובץ.

הפעולות שהמשתמש יכול לבצע על קבצים באמצעות לחיצה ימינה על תיקיות הם:

1. לפתוח
2. למחוק
3. לשנות שם
4. לדחוס לקובץ

את הקבצים ששיתפו איתך ניתן לראות על ידי לחיצה על הלוגו של ה3 אנשים, לכל משתמש שמשתף איתך קובץ יש תיקייה משלו ששם אפשר לראות את הקבצים.  
  
מעל הלוגו של האנשים ניתן לראות את תמונת הפרופיל של המשתמש, על ידי לחיצה על תמונת הפרופיל רואים את הגדרות המשתמש, כמו שם המשתמש המייל ותמונת הפרופיל. ניתן לשנות את ההגדרות באמצעות הכפתורים ואת הגדרות תמונת הפרופיל ניתן לשנות גם באמצעות לחיצה ימנית על תמונת הפרופיל.

# **פרק ח - 'goldrive' – רפלקציה**

**הפרויקט היה חוויה מהנה עם הרבה עליות ומורדות, התמודדות עם קשיים ועם לוחות זמנים. מאוד נהניתי לעבוד על הפרויקט בכיתה באווירה טובה ותומכת שאפשר לגשת אליה לעזרה.  
הקשיים שהיו לי בפרויקט היו בעיקר כתיבת תיק הפרויקט שלקח זמן רב ומחשבה לעומק על תכנון הקוד עוד לפני שאני בכלל נגש לכתוב אותו, קשיים נוספים היו לי בכתיבת הקוד בייחוד באופציה לערוך קובץ מתוך התוכנה, הקושי בזה היה בעיקר ההתמודדות עם פרוססים, איך פותחים אותם איך קוראים ואיך בודקים מתי הקובץ נפתח ומתי המשתמש עורך אותו, אני חושב שהתמודדתי טוב עם כל הקשיים שלי ועבדתי בצורה טובה ויעילה.**

**לדעתי הייתי יכול להשתפר בכך שאם הייתי יושב לעבוד יותר ברציפות וללא הפסקות הייתי יכול לסיים את הפרויקט יותר מוקדם ופחות בלחץ של זמן.**

**אני שמח מאוד עם התוצר הסופי שלי וחושב שהוספתי הרבה פיצ'רים גם שלא היו כתובים בתיק הפרויקט המקורי שלי שלדעתי נתנו חוויה יותר טובה למשתמש והוסיפו הרגשה של פרויקט יותר רציני.**

**המון תודה למרי על העזרה במהלך כל הפרויקט שהייתה שם לעזור ולתמוך ולחברי הכיתה שתמיד היה שיח עוזר ומפתח.**

# **פרק ט - 'goldrive' – ביבליוגרפיה**

**השתמשתי כדי ללמוד יותר על החבילה** wx **-  
 https://wxpython.org/index.html**

**השתמשתי כדי ללמוד להשתמש ב**grid sizer **- https://docs.wxpython.org/wx.GridSizer.html**

**השתמשתי כדי ללמוד איך לגרור אובייקטים ב -** wx python **https://www.tutorialspoint.com/wxpython/wxpython\_drag\_and\_drop.htm** -

**השתמשתי כדי ללמוד איך לדחוס תיקייה לקובץ - https://stackoverflow.com/questions/42214376/zip-single-file**

**השתמשתי כדי לשים תמונה כמסך רקע - https://www.blog.pythonlibrary.org/2010/03/18/wxpython-putting-a-background-image-on-a-panel/**

**השתמשתי כדי לשלוח מייל דרך התוכנה -   
https://realpython.com/python-send-email/**

**השתמשתי כדי ללמוד איך לקרוא את ה**pid **- https://www.geeksforgeeks.org/python-os-getpid-method/**

**השתמשתי כדי ללמוד איך להשתמש בפאנל שאפשר לגלול בו - https://docs.wxpython.org/wx.lib.scrolledpanel.html**

**השתמשתי כדי ללמוד איך להשתמש בפאנל שאפשר לגלול בו - https://stackoverflow.com/questions/21377946/creating-a-scrolling-panel-in-wxpython**

**השתמשתי כדי לשים תמונה לתוכנה - https://stackoverflow.com/questions/59933833/wxpython-application-icon**

# **נספחים**

clientComm:

**import** socket  
**import** threading  
**import** sys  
**import** queue  
**import** encryption  
**from** settings **import** CurrentSettings **as** Settings  
**import** clientProtocol  
**import** wx  
**from** pubsub **import** pub  
  
  
**class** ClientComm:  
 **def** \_\_init\_\_(self, server\_ip, port, recv\_q, msg\_len\_bytes):  
 *"""  
 init of clientComm object* **:param** *server\_ip: ip of the server* **:param** *port: port of the server* **:param** *recv\_q: msgs queue* **:param** *msg\_len\_bytes: how many bytes to fill to  
 """* self.server\_ip = server\_ip  
 self.port = port  
 self.recvQ = recv\_q  
 self.socket = socket.socket()  
 self.msgLenBytes = msg\_len\_bytes  
 self.enc\_obj = **None** threading.Thread(target=self.\_main\_loop).start()  
  
 **def** \_main\_loop(self):  
 *"""  
 main loop of the server* **:return***: receives data and puts in the q  
 """* **try**:  
 self.socket.connect((self.server\_ip, self.port))  
 **except** Exception **as** e:  
 sys.exit(**"Server is closed try again later, "** + str(e))  
  
 **else**:  
 *# changing key* self.enc\_obj = self.\_change\_key()  
  
 **while True**:  
 **try**:  
 leng = self.socket.recv(self.msgLenBytes).decode()  
 data = self.socket.recv(int(leng))  
 **except** Exception **as** e:  
 sys.exit(**"Server is closed try again later"**)  
  
 decrypted\_data = self.enc\_obj.dec\_msg(data)  
  
 *# is it the default port* **if** self.port == Settings.SERVERPORT:  
 self.recvQ.put(decrypted\_data)  
 **else**:  
 opcode, params = clientProtocol.unpack\_message(decrypted\_data)  
  
 **if** opcode == **"04" or** opcode == **"11" or** opcode == **"20" or** opcode == **"21"**:  
 **if** params[0] == **"0"**:  
 params = (opcode, \*params[1:])  
 self.\_recv\_file(params)  
 **else**:  
 self.recvQ.put(decrypted\_data)  
 **else**:  
 self.recvQ.put(decrypted\_data)  
  
 **def** \_change\_key(self):  
 *"""  
 swaps symmetric key with server  
 """* privateA, a = encryption.get\_dh\_factor()  
 **try**:  
 b = int(self.socket.recv(10).decode())  
 self.socket.send((str(a).zfill(10)).encode())  
  
 **except** Exception **as** e:  
 print(**"in change key "**, str(e))  
  
 **else**:  
 crypto = encryption.create\_symmetry\_key(privateA, b)  
 print(**"completed change key"**)  
  
 **return** crypto  
  
 **def** send(self, msg):  
 *"""* **:param** *msg: msg to send* **:return***: sends the message to the server  
 """* encrypt\_data = self.enc\_obj.enc\_msg(msg)  
  
 *# the length of the encrypted message* data\_len = str(len(encrypt\_data)).zfill(self.msgLenBytes).encode()  
 **try**:  
 self.socket.send(data\_len + encrypt\_data)  
 **except** Exception **as** e:  
 sys.exit(**"Server is closed try again later"**)  
  
 **def** \_close(self):  
 *"""  
 closes the client socket  
 """* self.socket.close()  
  
 **def** \_recv\_file(self, params):  
 *"""* **:param** *params: the params to use* **:return***: receives file and put in the q accordingly  
 """* opcode = params[0]  
 path, selected\_path, Type, email = **""**, **""**, **""**, **""** *# showing the progress bar* **if** opcode == **"04"**:  
 file\_len = params[1]  
 wx.CallAfter(pub.sendMessage, **"startBar"**, name=path.split(**'/'**)[-1], opcode=opcode)  
 **elif** opcode == **"11"**:  
 file\_len, path, selected\_path = params[1:]  
 wx.CallAfter(pub.sendMessage, **"startBar"**, name=path.split(**'/'**)[-1], opcode=opcode)  
 **elif** opcode == **"21"**:  
 file\_len, email = params[1:]  
 **else**:  
 file\_len, Type = params[1:]  
  
 data = bytearray()  
 file\_len = int(file\_len)  
  
 **try**:  
 **while** len(data) < file\_len:  
 slices = file\_len - len(data)  
  
 **if** slices > 1024:  
 **if** opcode == **"11" or** opcode == **"04"**:  
 *# changing the progress in the progress bar* wx.CallAfter(pub.sendMessage, **"changeProgress"**,  
 percent=int((len(data) / file\_len) \* 100), opcode=opcode)  
 data.extend(self.socket.recv(1024))  
 **else**:  
 data.extend(self.socket.recv(slices))  
 **break  
  
 except** Exception **as** e:  
 **if** opcode == **"04"**:  
 self.recvQ.put((**"04"**, **'1'**, **None**))  
 **elif** opcode == **"11"**:  
 self.recvQ.put((**"11"**, **'1'**, path, selected\_path, **None**))  
 **elif** opcode == **"21"**:  
 self.recvQ.put((**"21"**, **None**, **None**))  
 **else**:  
 self.recvQ.put((**"20"**, **'1'**, Type, **None**))  
  
 print(**"main server in recv file comm "**, str(e))  
 self.\_close()  
 **else**:  
 **if** opcode == **"04"**:  
 wx.CallAfter(pub.sendMessage, **"changeProgress"**,  
 percent=int((len(data) / file\_len) \* 100), opcode=opcode)  
 self.recvQ.put((**"04"**, self.enc\_obj.dec\_msg(data)))  
 **elif** opcode == **"11"**:  
 wx.CallAfter(pub.sendMessage, **"changeProgress"**,  
 percent=int((len(data) / file\_len) \* 100), opcode=opcode)  
 self.recvQ.put((**"11"**, **'0'**, path, selected\_path, self.enc\_obj.dec\_msg(data)))  
 **elif** opcode == **"21"**:  
 self.recvQ.put((**"21"**, email, self.enc\_obj.dec\_msg(data)))  
  
 **else**:  
 self.recvQ.put((**"20"**, **'0'**, Type, self.enc\_obj.dec\_msg(data)))  
  
 **def** send\_file(self, opcode, path, currPath):  
 *"""* **:param** *opcode: opcode of msg to send* **:param** *path: path of file to send* **:param** *currPath: path in the server* **:return***: sends file to the server  
 """* **try**:  
 **with** open(path, **'rb'**) **as** f:  
 data = f.read()  
 **except** Exception **as** e:  
 print(str(e))  
  
 **else**:  
 cryptFile = self.enc\_obj.enc\_msg(data)  
 **if** opcode == 12:  
 msg = clientProtocol.pack\_upload\_file\_request(**f"{**currPath**}/{**path.split(**'/'**)[-1]**}"**.lstrip(**'/'**),  
 len(cryptFile))  
 **else**:  
 msg = clientProtocol.pack\_change\_photo\_request(currPath, len(cryptFile))  
  
 self.send(msg)  
 **try**:  
 self.socket.send(cryptFile)  
 **except** Exception **as** e:  
 print(str(e))  
  
 **def** did\_change\_work(self):  
 *"""* **:return***: did the change key work  
 """* **return** self.enc\_obj **is not None  
  
  
if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 msgQ = queue.Queue()  
 server = ClientComm(Settings.SERVERIP, Settings.SERVERPORT, msgQ, 2)  
  
 **while not** server.enc\_obj:  
 **pass** server.send(clientProtocol.pack\_register\_request(**"lior123"**, **"check123"**, **"reefg19@gmail.com"**))  
  
 **while True**:  
 data = msgQ.get()  
 print(data)

clientLogic:  
**import** os.path  
**import** queue  
**import** clientComm  
**import** clientProtocol  
**from** settings **import** CurrentSettings **as** Settings  
**import** threading  
**from** Graphics.mainPanel **import** MainFrame  
**from** pubsub **import** pub  
**import** wx  
**import** secrets  
**import** shutil  
**import** monitorFile  
**import** time  
**import** sys  
  
  
**def** main\_loop():  
 *"""  
 main loop of the server  
 """* msg\_q = queue.Queue()  
  
 client\_comm = clientComm.ClientComm(Settings.SERVERIP, Settings.SERVERPORT, msg\_q, 4)  
  
 starting\_time = time.time()  
  
 *# giving the client 5 seconds to try connecting to the server* **while** time.time() - starting\_time < 5 **and not** client\_comm.did\_change\_work():  
 **pass** change\_keys\_success = client\_comm.did\_change\_work()  
  
 **if** change\_keys\_success:  
 threading.Thread(target=\_handle\_messages, args=(msg\_q,)).start()  
  
 app = wx.App()  
 MainFrame(client\_comm)  
  
 app.MainLoop()  
 **else**:  
 sys.exit(**"Server is closed "**)  
  
  
**def** \_handle\_messages(msg\_q):  
 *"""  
 handles the messages from queue* **:param** *msg\_q: the queue of the messages to handle* **:return***: activates the according function according to the opcode  
 """* recv\_commands = {**"01"**: \_show\_register\_dialog, **"02"**: \_show\_login\_dialog, **"03"**: \_handle\_send\_files,  
 **"05"**: \_handle\_change\_email, **"06"**: \_handle\_change\_password, **"08"**: \_handle\_rename\_file,  
 **"09"**: \_handle\_share\_file, **"10"**: \_handle\_delete\_file, **"13"**: \_handle\_create\_dir,  
 **"14"**: \_handle\_add\_shared\_file, **"16"**: \_handle\_files\_port, **"17"**: \_handle\_username\_check,  
 **"18"**: \_handle\_move\_file, **"19"**: \_handle\_paste\_file, **"23"**: \_handle\_login,  
 **"24"**: \_handle\_register, **"25"**: \_handle\_code\_check, **"26"**: \_handle\_forgot\_password,  
 **"27"**: \_handle\_zip\_file}  
  
 **while True**:  
 data = msg\_q.get()  
 protocol\_num, params = clientProtocol.unpack\_message(data)  
  
 **if** protocol\_num == **"03"**:  
 *# since \* breaks the list down and we want the whole list* \_handle\_send\_files(params)  
 **else**:  
 recv\_commands[protocol\_num](\*params)  
  
  
**def** \_handle\_files\_port(files\_port):  
 *"""  
 connects to the files port* **:param** *files\_port: the port of the files port to connect to* **:return***: connects to the server to the files port  
 """* files\_q = queue.Queue()  
 files\_comm = clientComm.ClientComm(Settings.SERVERIP, int(files\_port), files\_q, 10)  
 threading.Thread(target=\_handle\_files, args=(files\_comm, files\_q,)).start()  
 wx.CallAfter(pub.sendMessage, **"updateFileComm"**, file\_comm=files\_comm)  
  
  
**def** \_handle\_files(files\_comm, files\_q):  
 *"""  
 handles messages from the file q* **:param** *files\_comm: the comm object of the client which handles the uploads, downloads etc.* **:param** *files\_q: the queue of message to handle of the files port* **:return***: activates the according function according to the opcode  
 """* recv\_commands = {**"04"**: \_handle\_change\_photo, **"11"**: \_handle\_download\_file, **"12"**: \_handle\_upload\_file,  
 **"20"**: \_handle\_open\_file, **"21"**: \_handle\_get\_details}  
  
 **while True**:  
 data = files\_q.get()  
 **if** data[0] == **"04" or** data[0] == **"11" or** data[0] == **"20" or** data[0] == **"21"**:  
 protocol\_num, \*params = data  
 **if** data[0] == **"20"**:  
 params.insert(0, files\_comm)  
 **else**:  
 protocol\_num, params = clientProtocol.unpack\_message(data)  
  
 recv\_commands[protocol\_num](\*params)  
  
  
**def** \_show\_register\_dialog(status, username, password, email):  
 *"""* **:param** *status: was the verify email sent successfully* **:param** *username: username wanting to register* **:param** *password: password wanting to register* **:param** *email: email wanting to register* **:return***: either showing the verify code dialog or showing the error message  
 """* **if** status == **"0"**:  
 *# show dialog to enter verify code* wx.CallAfter(pub.sendMessage, **"showRegisterDialog"**, username=username, password=password, email=email)  
 **elif** status == **"1"**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"User already exists."**, title=**"Error"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Please enter a valid username and password."**, title=**"Error"**)  
  
  
**def** \_handle\_register(status):  
 *"""* **:param** *status: was the register successful* **:return***: showing the correct msg to the screen  
 """* **if** status == **"0"**:  
 *# registered successfully* wx.CallAfter(pub.sendMessage, **"registerOk"**)  
 **elif** status == **"1"**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"User already exists."**, title=**"Error"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Wrong code entered."**, title=**"Error"**)  
  
  
**def** \_show\_login\_dialog(status, email):  
 *"""* **:param** *status: was the verify email sent successfully* **:param** *email: the email sent the verify code to* **:return***: either showing the verify code dialog or showing the error message  
 """* **if** status == **"0"**:  
 *# show verify code dialog* wx.CallAfter(pub.sendMessage, **"showLoginDialog"**, email=email)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Wrong username or password entered"**, title=**"Error"**)  
  
  
**def** \_handle\_login(status):  
 *"""* **:param** *status: was the login successful* **:return***: either showing the files of the user or showing error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"loginOk"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Wrong verification code."**, title=**"Error"**)  
  
  
**def** \_handle\_delete\_file(status):  
 *"""* **:param** *status: was the file deleted successfully* **:return***: tells graphic if to delete from the files of the user or to show error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"deleteOk"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't delete file."**, title=**"Error"**)  
  
  
**def** \_handle\_rename\_file(status, new\_name):  
 *"""* **:param** *status: was the rename successful* **:param** *new\_name: new name of the file* **:return***: tells graphic if to rename the file from the files of the user or to show error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"renameOk"**, new\_name=new\_name)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't rename file."**, title=**"Error"**)  
  
  
**def** \_handle\_download\_file(status, path, selected\_path, data):  
 *"""* **:param** *status: was the download successful* **:param** *path: path to download to* **:param** *selected\_path: path of file in the server* **:param** *data: data of the file downloaded* **:return***: downloads the file  
 """* file\_name = path.split(**'/'**)[-1]  
  
 **if** status == **"0"**:  
 **try**:  
 *# writing the file* **with** open(selected\_path, **'wb' if** type(data) **is** bytes **else 'w'**) **as** f:  
 f.write(data)  
  
 *# showing successfully download message* wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**f"Downloaded {**file\_name**} to {**selected\_path**} successfully."**,  
 title=**"Success"**)  
  
 **except** Exception **as** e:  
 *# showing error in download message* wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Download failed."**, title=**"Error"**)  
 print(str(e))  
  
 **else**:  
 *# showing error in download message* wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Download failed."**, title=**"Error"**)  
  
  
**def** \_handle\_open\_file(file\_comm, status, server\_path, data):  
 *"""  
 :param file\_comm: the file comm object  
 :param status: was the download successful  
 :param server\_path: path of the file in the server  
 :param data: data of the file  
 :return: opens the file for edit  
 """* file\_name = server\_path.split(**'/'**)[-1]  
 server\_path = **'/'**.join(server\_path.split(**'/'**)[:-1])  
  
 **if** status == **"0"**:  
 *# creating random name* ascii\_values = list(range(65, 91)) + list(range(97, 123)) + list(range(48, 58))  
  
 cwd = os.getcwd().replace(**'\\'**, **'/'**)  
 **while True**:  
 random\_name = **''**.join(chr(secrets.choice(ascii\_values)) **for** \_ **in** range(16))  
 path = **f"{**cwd**}/{**random\_name**}"  
 if not** os.path.isdir(path):  
 **break  
  
 try**:  
 *# creating temp file* os.mkdir(path)  
 file\_path = **f"{**path**}/{**file\_name**}"  
 with** open(file\_path, **'wb' if** type(data) **is** bytes **else 'w'**) **as** f:  
 f.write(data)  
  
 **except** Exception **as** e:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't open file."**, title=**"Error"**)  
 print(str(e))  
 **else**:  
 change\_flag = **False** monitor\_q = queue.Queue()  
  
 *# has the file been closed yet* is\_alive\_thread = threading.Thread(target=monitorFile.wait\_until, args=(file\_path, monitor\_q))  
 is\_alive\_thread.start()  
  
 *# were there any changes made to the file* monitor\_thread = threading.Thread(target=monitorFile.monitor, args=(path, monitor\_q))  
 monitor\_thread.start()  
  
 **while True**:  
 data = monitor\_q.get()  
 *# if closed the file* **if** data == **"Finished"**:  
 **break** *# if made changed to the file* **elif** data == **"Changed"**:  
 change\_flag = **True** *# if any changes were made update the server to change the file* **if** change\_flag:  
 file\_comm.send\_file(12, file\_path, server\_path)  
  
 *# deleting the temp file* **try**:  
 shutil.rmtree(path)  
 **except** Exception **as** e:  
 print(str(e))  
  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't open file."**, title=**"Error"**)  
  
  
**def** \_handle\_upload\_file(status, path):  
 *"""  
 :param status: was the upload successful  
 :param path: path of the file uploaded  
 :return: tells graphics to add the file to the user files or shows error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"uploadOk"**, path=path)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't upload file."**, title=**"Error"**)  
  
  
**def** \_handle\_create\_dir(status):  
 *"""  
 :param status: was the creating of the directory successful  
 :return: tells graphics to add the older to the user files or shows error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"createOk"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't create folder."**, title=**"Error"**)  
  
  
**def** \_handle\_share\_file(status):  
 *"""  
 :param status: was the share successful  
 :return: tells graphics to show wether the share was successful or not  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Shared file successfully."**, title=**"Success"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"User doesn't exist."**, title=**"Error"**)  
  
  
**def** \_handle\_add\_shared\_file(path):  
 *"""  
 :param path: path of file shared  
 :return: adding the file shared  
 """* wx.CallAfter(pub.sendMessage, **"addFile"**, path=path)  
  
  
**def** \_handle\_move\_file(status, path):  
 *"""  
 :param status: was the move successful  
 :param path: where to move the file to  
 :return: tells graphics to move file in the user files or show error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"moveFile"**, path=path)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't move file."**, title=**"Error"**)  
  
  
**def** \_handle\_paste\_file(status):  
 *"""  
 :param status: was the paste successful  
 :return: tells graphics to update user files or show error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"pasteFile"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't paste file"**, title=**"Error"**)  
  
  
**def** \_handle\_get\_details(email, photo):  
 *"""* **:param** *email: email of user* **:param** *photo: data of photo of user* **:return***:  
 """* **if** email:  
 wx.CallAfter(pub.sendMessage, **"detailsOk"**, email=email, photo=photo)  
  
  
**def** \_handle\_change\_email(status, email):  
 *"""* **:param** *status: was the email change successful* **:param** *email: email to change to* **:return***: changes email in graphics or showing error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"changeEmailOk"**, email=email)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't change email, try again."**, title=**"Error"**)  
  
  
**def** \_handle\_change\_password(status):  
 *"""* **:param** *status: was the password change successful* **:return***: show message whether the change was successful or not  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Password changed successfully."**, title=**"Success"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't change password, try again."**, title=**"Error"**)  
  
  
**def** \_handle\_change\_photo(data):  
 *"""  
 :param data: data of photo to change to  
 :return: changing the photo data in the graphics or showing error message  
 """* **if** data:  
 wx.CallAfter(pub.sendMessage, **"changePhotoOk"**, data=data)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't change photo, try again."**, title=**"Error"**)  
  
  
**def** \_handle\_username\_check(status, email):  
 *"""  
 :param status: does the user exist  
 :param email: email of user if he exists None if not  
 :return: showing code verify dialog or showing error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"forgotPassEmailOk"**, email=email)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"User doesn't exist."**, title=**"Error"**)  
  
  
**def** \_handle\_code\_check(status):  
 *"""  
 :param status: was the code entered correct  
 :return: showing forgot password dialog or showing error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"showForgotPassDialog"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Wrong code entered."**, title=**"Error"**)  
  
  
**def** \_handle\_forgot\_password(status):  
 *"""  
 :param status: was the password changed successfully  
 :return: showing message whether it was successful or not  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Password changed successfully."**, title=**"Success"**)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't change password."**, title=**"Error"**)  
  
  
**def** \_handle\_zip\_file(status, zip\_name):  
 *"""  
 :param status: was the zip file created successfully or not  
 :param zip\_name: the name of the zip file created  
 :return: telling graphics to add the zip to the user files or show error message  
 """* **if** status == **"0"**:  
 wx.CallAfter(pub.sendMessage, **"zipFileOk"**, path=zip\_name)  
 **else**:  
 wx.CallAfter(pub.sendMessage, **"showPopUp"**, text=**"Couldn't zip file."**, title=**"Error"**)  
  
  
**def** \_handle\_send\_files(branches):  
 *"""  
 :param branches: files of the user  
 :return: updating the files of the user  
 """* wx.CallAfter(pub.sendMessage, **"filesOk"**, branches=branches)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main\_loop()

clientProtocol:  
**def** unpack\_message(msg):  
 *"""* **:param** *msg: message to unpack* **:return***: tuple with opcode and variables  
 """* opcode = msg[:2]  
 msg = msg[2:]  
 data\_from\_msg = []  
 **if** opcode == **"03"**:  
 data\_from\_msg = unpack\_files\_message(msg)  
  
 **else**:  
 **while** len(msg) > 0:  
 **try**:  
 data, msg = get\_data\_from\_string(msg, 2)  
 data\_from\_msg.append(data)  
 **except** Exception **as** e:  
 print(**"in unpack\_message -"**, str(e))  
 opcode = **None  
 break  
  
 return** opcode, data\_from\_msg  
  
  
**def** get\_data\_from\_string(line, length):  
 *"""* **:param** *line: line to slice* **:param** *length: length to slice* **:return***:  
 """  
 # getting the data from the length of the string* data = line[length:length + int(line[:length])]  
  
 *# substring* line = line[length + int(line[:length])::]  
  
 **return** data, line  
  
  
**def** unpack\_files\_message(msg):  
 *"""* **:param** *msg: the files message* **:return***: list of branches each branch has its directories and files  
 """* branches = []  
  
 *# until last element since its blank* lines = msg.split(**'\n'**)[:-1]  
  
 **for** line **in** lines:  
 name\_to\_add, line = get\_data\_from\_string(line, 2)  
 dirs, line = get\_data\_from\_string(line, 4)  
 files, line = get\_data\_from\_string(line, 4)  
  
 *# joined directories with !* dirs = dirs.split(**'!'**)  
  
 *# joined files with @* files = files.split(**'@'**)  
  
 *# adding to list of branches* branches.append((name\_to\_add, dirs, files))  
  
 **return** branches  
  
  
**def** pack\_register\_request(username, password, email):  
 *"""* **:param** *username: username* **:param** *password: password* **:param** *email: email* **:return***: message built by protocol  
 """* **return f"01{**str(len(username)).zfill(2)**}{**username**}{**str(len(password)).zfill(2)**}{**password**}"** \  
 **f"{**str(len(email)).zfill(2)**}{**email**}"  
  
  
def** pack\_login\_request(username, password):  
 *"""* **:param** *username: username* **:param** *password: password* **:return***: message built py protocol  
 """* **return f"02{**str(len(username)).zfill(2)**}{**username**}{**str(len(password)).zfill(2)**}{**password**}"  
  
  
def** pack\_verify\_check\_request(email, code):  
 *"""* **:param** *email: email to verify* **:param** *code: code sent to email* **:return***: message built py protocol  
 """* **return f"15{**str(len(email)).zfill(2)**}{**email**}{**str(len(str(code))).zfill(2)**}{**code**}"  
  
  
def** pack\_send\_verify\_request(email):  
 *"""* **:param** *email: email* **:return***: message built py protocol  
 """* **return f"07{**str(len(email)).zfill(2)**}{**email**}"  
  
  
def** pack\_download\_file\_request(path, selected\_path):  
 *"""* **:param** *path: path of file to download* **:param** *selected\_path: path of where to download the file to* **:return***: message built py protocol  
 """* **return f"11{**str(len(path)).zfill(2)**}{**path**}{**str(len(selected\_path)).zfill(2)**}{**selected\_path**}"  
  
  
def** pack\_upload\_file\_request(upload\_path, data\_len):  
 *"""* **:param** *upload\_path: path to upload to including file name* **:param** *data\_len: the length of the data* **:return***: message built py protocol  
 """* **return f"12{**str(len(upload\_path)).zfill(2)**}{**upload\_path**}{**str(len(str(data\_len))).zfill(2)**}{**data\_len**}"  
  
  
def** pack\_create\_folder\_request(path):  
 *"""* **:param** *path: path of where to create the folder* **:return***: message built py protocol  
 """* **return f"13{**str(len(path)).zfill(2)**}{**path**}"  
  
  
def** pack\_delete\_request(path):  
 **"""  
 :param path: path of file to delete  
 :return: message built py protocol  
 """""  
 return f"10{**str(len(path)).zfill(2)**}{**path**}"  
  
  
def** pack\_rename\_file\_request(path, new\_name):  
 *"""* **:param** *path: path of file to rename* **:param** *new\_name: new name of file* **:return***: message built py protocol  
 """* **return f"08{**str(len(path)).zfill(2)**}{**path**}{**str(len(new\_name)).zfill(2)**}{**new\_name**}"  
  
  
def** pack\_share\_request(path, username):  
 *"""* **:param** *path: file to share path* **:param** *username: username of person to share* **:return***: message built py protocol  
 """* **return f"09{**str(len(path)).zfill(2)**}{**path**}{**str(len(username)).zfill(2)**}{**username**}"  
  
  
def** pack\_change\_photo\_request(username, file\_len):  
 *"""* **:param** *username: username of user to change photo of* **:param** *file\_len: length of the photo* **:return***: message built py protocol  
 """* **return f"04{**str(len(str(username))).zfill(2)**}{**username**}{**str(len(str(file\_len))).zfill(2)**}{**file\_len**}"  
  
  
def** pack\_change\_password\_request(username, old\_password, new\_password, confirmed\_password):  
 *"""* **:param** *username: username of user* **:param** *old\_password: password* **:param** *new\_password: new password of user* **:param** *confirmed\_password: confirmation of the new password* **:return***: message built py protocol  
 """* **return** (**f"06{**str(len(str(username))).zfill(2)**}{**username**}{**str(len(str(old\_password))).zfill(2)**}{**old\_password**}"  
 f"{**str(len(str(new\_password))).zfill(2)**}{**new\_password**}"  
 f"{**str(len(str(confirmed\_password))).zfill(2)**}{**confirmed\_password**}"**)  
  
  
**def** pack\_change\_email\_request(username, email, code):  
 *"""* **:param** *username: username of user* **:param** *email: email* **:param** *code: the code sent to the user* **:return***: message built py protocol  
 """* **return** (**f"05{**str(len(str(username))).zfill(2)**}{**username**}{**str(len(str(email))).zfill(2)**}{**email**}"  
 f"{**str(len(str(code))).zfill(2)**}{**code**}"**)  
  
  
**def** pack\_send\_email\_request(username):  
 *"""* **:param** *username: username to send mail to  
 :return: message built by protocol  
 """* **return f"17{**str(len(str(username))).zfill(2)**}{**username**}"  
  
  
def** pack\_move\_file\_folder\_request(src, dst):  
 *"""* **:param** *src: path of file to move* **:param** *dst: path of destination* **:return***: message built by protocol  
 """* **return f"18{**str(len(str(src))).zfill(2)**}{**src**}{**str(len(str(dst))).zfill(2)**}{**dst**}"  
  
  
def** pack\_paste\_file\_request(src, dst):  
 *"""* **:param** *src: path of file to paste* **:param** *dst: path of where to paste* **:return***: message built by protocol  
 """* **return f"19{**str(len(str(src))).zfill(2)**}{**src**}{**str(len(str(dst))).zfill(2)**}{**dst**}"  
  
  
def** pack\_open\_file\_request(path):  
 *"""* **:param** *path: path of file to open* **:return***: message built by protocol  
 """* **return f"20{**str(len(str(path))).zfill(2)**}{**path**}"  
  
  
def** pack\_get\_details\_request(username):  
 *"""  
 :param username: username to get details of* **:return***: message built by protocol  
 """* **return f"21{**str(len(str(username))).zfill(2)**}{**username**}"  
  
  
def** pack\_delete\_profile\_photo\_request(username):  
 *"""* **:param** *username: username of user to delete the profile photo of* **:return***: message built by protocol  
 """* **return f"22{**str(len(str(username))).zfill(2)**}{**username**}"  
  
  
def** pack\_verify\_login\_email\_request(email, code, dont\_ask\_again, username):  
 *"""* **:param** *email: email* **:param** *code: code sent to email* **:param** *dont\_ask\_again: does the user want to verify again on this computer* **:param** *username: username of user* **:return***: message built by protocol  
 """* **return** (**f"23{**str(len(str(email))).zfill(2)**}{**email**}{**str(len(str(code))).zfill(2)**}{**code**}"  
 f"{**str(len(str(dont\_ask\_again))).zfill(2)**}{**dont\_ask\_again**}{**str(len(str(username))).zfill(2)**}{**username**}"**)  
  
  
**def** pack\_verify\_register\_email\_request(username, password, email, code, dont\_ask\_again):  
 *"""* **:param** *username: username to register* **:param** *password: password to register* **:param** *email: email to register* **:param** *code: code entered by user* **:param** *dont\_ask\_again: does the user want to confirm mail again on this computer* **:return***: message built by protocol  
 """* **return f"24{**str(len(str(username))).zfill(2)**}{**username**}{**str(len(str(password))).zfill(2)**}{**password**}"** \  
 **f"{**str(len(str(email))).zfill(2)**}{**email**}{**str(len(str(code))).zfill(2)**}{**code**}"** \  
 **f"{**str(len(str(dont\_ask\_again))).zfill(2)**}{**dont\_ask\_again**}"  
  
  
def** pack\_check\_code\_request(email, code):  
 *"""* **:param** *email: email code was sent to* **:param** *code: the code the user entered* **:return***: message built by protocol  
 """* **return f"25{**str(len(str(email))).zfill(2)**}{**email**}{**str(len(str(code))).zfill(2)**}{**code**}"  
  
  
def** pack\_forgot\_password\_request(username, password, confirmed\_pass):  
 *"""* **:param** *username: username of user* **:param** *password: password to change to* **:param** *confirmed\_pass: confirm of password to change to* **:return***:  
 """* **return f"26{**str(len(str(username))).zfill(2)**}{**username**}{**str(len(str(password))).zfill(2)**}{**password**}"** \  
 **f"{**str(len(str(confirmed\_pass))).zfill(2)**}{**confirmed\_pass**}"  
  
  
def** pack\_zip\_folder\_request(folder\_path):  
 *"""* **:param** *folder\_path: path of folder to zip* **:return***: message built by protocol  
 """* **return f"27{**str(len(str(folder\_path))).zfill(2)**}{**folder\_path**}"  
  
  
def** disconnect\_alert():  
 *"""* **:return***:  
 """* **return f"28"  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 print(unpack\_files\_message(**"000007folder10009fil1.docx\n07folder10008checking0000\n16folder1\\checking00000000"**))  
  
 print(pack\_register\_request(**"reef"**, **"123"**, **"reefg19@gmail.com"**))  
 print(unpack\_message(**"0104reef0312317reefg19@gmail.com"**))  
 print(pack\_login\_request(**"reef"**, **"123"**))  
  
 print(pack\_get\_details\_request(**"reef"**))

customMenusAndDialogs:  
**from** settings **import** CurrentSettings **as** Settings  
**import** wx  
  
  
**class** FileMenuFeatures(wx.Menu):  
 **def** \_\_init\_\_(self, parent, name):  
 *"""  
 :param parent: parent panel  
 :param name: name of what the user clicked on  
 """* super(FileMenuFeatures, self).\_\_init\_\_()  
  
 image\_types = [**"apng"**, **"avif"**, **"gif"**, **"jpg"**, **"jpeg"**, **"jfif"**, **"pjpeg"**, **"pjp"**, **"png"**, **"svg"**, **"webp"**, **"bmp"**, **"ico"**,  
 **"cur"**, **"tif"**, **"tiff"**]  
  
 self.parent = parent  
 self.name = name  
  
 static\_text\_item = wx.MenuItem(self, 0, name)  
 *# Make it non-selectable* static\_text\_item.Enable(**False**)  
 self.Append(static\_text\_item)  
  
 self.AppendSeparator()  
  
 self.Append(wx.MenuItem(self, -1, **'Delete file'**))  
 self.Append(wx.MenuItem(self, -1, **'Rename file'**))  
 self.Append(wx.MenuItem(self, -1, **'Download file'**))  
 self.Append(wx.MenuItem(self, -1, **'Share file'**))  
 self.Append(wx.MenuItem(self, -1, **'Copy file'**))  
  
 view\_text = **"Edit file"  
 if** self.name.split(**'.'**)[-1] **in** image\_types:  
 view\_text = **"Open file"** self.Append(wx.MenuItem(self, -1, view\_text))  
  
 self.AppendSeparator()  
  
 self.Append(wx.MenuItem(self, -1, **"Cancel"**))  
  
 *# Bind the menu item click event* self.Bind(wx.EVT\_MENU, self.on\_menu\_item\_click)  
  
 **def** on\_menu\_item\_click(self, event):  
 *"""  
 :param event: event  
 :return: starts the correct function based on what the user clicked  
 """* clicked\_item\_id = event.GetId()  
 clicked\_item = self.FindItemById(clicked\_item\_id)  
  
 item = clicked\_item.GetItemLabelText()  
 **if** item == **"Delete file"**:  
 self.parent.delete\_file\_request(self.name)  
 **elif** item == **"Rename file"**:  
 self.parent.rename\_file\_request(self.name)  
 **elif** item == **"Download file"**:  
 self.parent.download\_file\_request(self.name)  
 **elif** item == **"Share file"**:  
 self.parent.share\_file\_request(self.name)  
 **elif** item == **"Copy file"**:  
 self.parent.copied\_file = **f"{**self.parent.curPath**}/{**self.name**}"**.lstrip(**'/'**)  
 **elif** item == **"Edit file" or** item == **"Open file"**:  
 self.parent.open\_file\_request(self.name)  
  
  
**class** FolderMenuFeatures(wx.Menu):  
 **def** \_\_init\_\_(self, parent, name):  
 *"""  
 :param parent: parent panel  
 :param name: name of what the user clicked on  
 """* super(FolderMenuFeatures, self).\_\_init\_\_()  
  
 self.parent = parent  
 self.name = name  
  
 static\_text\_item = wx.MenuItem(self, 0, name)  
 *# Make it non-selectable* static\_text\_item.Enable(**False**)  
 self.Append(static\_text\_item)  
  
 self.AppendSeparator()  
  
 self.Append(wx.MenuItem(self, -1, **'Open'**))  
 self.Append(wx.MenuItem(self, -1, **'Delete folder'**))  
 self.Append(wx.MenuItem(self, -1, **'Rename folder'**))  
 self.Append(wx.MenuItem(self, -1, **'Zip folder'**))  
  
 self.AppendSeparator()  
  
 self.Append(wx.MenuItem(self, -1, **"Cancel"**))  
  
 *# Bind the menu item click event* self.Bind(wx.EVT\_MENU, self.on\_menu\_item\_click)  
  
 **def** on\_menu\_item\_click(self, event):  
 *"""  
 :param event: event  
 :return: starts the according function to what the user pressed on  
 """* clicked\_item\_id = event.GetId()  
 clicked\_item = self.FindItemById(clicked\_item\_id)  
  
 item = clicked\_item.GetItemLabelText()  
 **if** item == **"Open"**:  
 self.parent.chose\_dir(self.name)  
 **elif** item == **"Delete folder"**:  
 self.parent.delete\_file\_request(self.name)  
 **elif** item == **"Rename folder"**:  
 self.parent.rename\_file\_request(self.name)  
 **elif** item == **"Zip folder"**:  
 self.parent.zip\_folder\_request(self.name)  
  
  
**class** UserMenuFeatures(wx.Menu):  
 **def** \_\_init\_\_(self, parent, show\_back):  
 *"""  
 :param parent: parent panel  
 :param show\_back: show go back flag  
 """* super(UserMenuFeatures, self).\_\_init\_\_()  
  
 self.parent = parent  
  
 self.Append(wx.MenuItem(self, -1, **'Upload file'**))  
 self.Append(wx.MenuItem(self, -1, **'Create new directory'**))  
 self.Append(wx.MenuItem(self, -1, **'Paste file'**))  
 self.back\_item = wx.MenuItem(self, -1, **'Go back'**)  
 self.Append(self.back\_item)  
 self.Enable(self.back\_item.GetId(), show\_back)  
  
 self.AppendSeparator()  
  
 self.Append(wx.MenuItem(self, -1, **"Cancel"**))  
  
 *# Bind the menu item click event* self.Bind(wx.EVT\_MENU, self.on\_menu\_item\_click)  
  
 **def** on\_menu\_item\_click(self, event):  
 *"""  
 :param event: event  
 :return: starts the according function to what the user pressed on  
 """* clicked\_item\_id = event.GetId()  
 clicked\_item = self.FindItemById(clicked\_item\_id)  
  
 item = clicked\_item.GetItemLabelText()  
 **if** item == **"Upload file"**:  
 self.parent.upload\_file\_request(wx.CommandEvent)  
 **elif** item == **"Create new directory"**:  
 self.parent.create\_dir\_request(wx.CommandEvent)  
 **elif** item == **"Paste file"**:  
 self.parent.paste\_file\_request(wx.CommandEvent)  
 **elif** item == **"Go back"**:  
 self.parent.back\_dir(wx.CommandEvent)  
  
  
**class** ProfileSettingsMenu(wx.Menu):  
 **def** \_\_init\_\_(self, parent):  
 *"""  
 :param parent: parent panel  
 """* super(ProfileSettingsMenu, self).\_\_init\_\_()  
  
 self.parent = parent  
  
 self.Append(wx.MenuItem(self, -1, **'Change profile photo'**))  
 self.Append(wx.MenuItem(self, -1, **'Clear profile photo'**))  
  
 self.AppendSeparator()  
  
 self.Append(wx.MenuItem(self, -1, **"Cancel"**))  
  
 *# Bind the menu item click event* self.Bind(wx.EVT\_MENU, self.on\_menu\_item\_click)  
  
 **def** on\_menu\_item\_click(self, event):  
 *"""  
 :param event: event  
 :return: strats the according function to what the user clicked on  
 """* item\_id = event.GetId()  
 clicked\_item = self.FindItemById(item\_id)  
  
 item = clicked\_item.GetItemLabelText()  
 **if** item == **"Change profile photo"**:  
 self.parent.change\_photo\_request(wx.CommandEvent)  
 **elif** item == **"Clear profile photo"**:  
 self.parent.delete\_photo\_request(wx.CommandEvent)  
  
  
**class** ChangePasswordDialog(wx.Dialog):  
 **def** \_\_init\_\_(self, parent, title):  
 *"""  
 :param parent: parent panel  
 :param title: title of the dialog  
 """* super(ChangePasswordDialog, self).\_\_init\_\_(parent, title=title, size=(388, 230), pos=(960-174, 540-115))  
  
 self.panel = wx.Panel(self)  
  
 self.is\_showing\_password = **False** hidden\_path = **f"{**Settings.USER\_FILES\_PATH**}\\hidden-eye.png"** open\_path = **f"{**Settings.USER\_FILES\_PATH**}\\open-eye.png"** self.hidden\_bitmap = wx.Bitmap(hidden\_path, wx.BITMAP\_TYPE\_ANY)  
 self.shown\_bitmap = wx.Bitmap(open\_path, wx.BITMAP\_TYPE\_ANY)  
  
 *# Create two text boxes* self.old\_password\_ctrl = wx.TextCtrl(self.panel, size=(308, -1), style=wx.TE\_PASSWORD)  
 self.new\_password\_ctrl = wx.TextCtrl(self.panel, size=(308, -1), style=wx.TE\_PASSWORD)  
 self.confirm\_password\_ctrl = wx.TextCtrl(self.panel, size=(308, -1), style=wx.TE\_PASSWORD)  
  
 *# Create OK and Cancel buttons* self.ok\_button = wx.Button(self.panel, label=**"OK"**)  
 self.cancel\_button = wx.Button(self.panel, label=**"Cancel"**)  
  
 *# Bind events to buttons* self.ok\_button.Bind(wx.EVT\_BUTTON, self.on\_ok)  
 self.cancel\_button.Bind(wx.EVT\_BUTTON, self.on\_cancel)  
  
 *# Set up the layout using a box sizer* sizer = wx.BoxSizer(wx.VERTICAL)  
 entries = wx.BoxSizer(wx.VERTICAL)  
 entries.Add(wx.StaticText(self.panel, label=**"Enter old password:"**), 0, wx.LEFT | wx.ALL)  
  
 old\_password\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 old\_password\_sizer.Add(self.old\_password\_ctrl, 0, wx.CENTER | wx.ALL, 5)  
 self.old\_pass\_icon = wx.StaticBitmap(self.panel, wx.ID\_ANY, self.hidden\_bitmap)  
 self.old\_pass\_icon.name = **"old"** old\_password\_sizer.Add(self.old\_pass\_icon)  
 entries.Add(old\_password\_sizer)  
  
 entries.AddSpacer(2)  
 entries.Add(wx.StaticText(self.panel, label=**"Enter new password:"**), 0, wx.LEFT | wx.ALL)  
  
 new\_password\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 new\_password\_sizer.Add(self.new\_password\_ctrl, 0, wx.CENTER | wx.ALL, 5)  
 self.new\_pass\_icon = wx.StaticBitmap(self.panel, wx.ID\_ANY, self.hidden\_bitmap)  
 self.new\_pass\_icon.name = **"new"** new\_password\_sizer.Add(self.new\_pass\_icon)  
 entries.Add(new\_password\_sizer)  
  
 entries.AddSpacer(2)  
 entries.Add(wx.StaticText(self.panel, label=**"Confirm password:"**), 0, wx.LEFT | wx.ALL)  
  
 con\_password\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 con\_password\_sizer.Add(self.confirm\_password\_ctrl, 0, wx.CENTER | wx.ALL, 5)  
 self.con\_pass\_icon = wx.StaticBitmap(self.panel, wx.ID\_ANY, self.hidden\_bitmap)  
 self.con\_pass\_icon.name = **"con"** con\_password\_sizer.Add(self.con\_pass\_icon)  
 entries.Add(con\_password\_sizer)  
  
 entries.AddSpacer(5)  
 sizer.Add(entries, 0, wx.CENTER | wx.ALL)  
  
 buttons\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 buttons\_sizer.Add(self.ok\_button, 0, wx.ALL)  
 buttons\_sizer.AddSpacer(10)  
 buttons\_sizer.Add(self.cancel\_button, 0, wx.ALL)  
 buttons\_sizer.AddSpacer(15)  
  
 sizer.Add(buttons\_sizer, 0, wx.ALIGN\_RIGHT)  
  
 self.old\_pass\_icon.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
 self.new\_pass\_icon.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
 self.con\_pass\_icon.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
  
 self.panel.SetSizerAndFit(sizer)  
  
 **def** change\_visibility(self, event):  
 *"""  
 :param event: event  
 :return: changes if you can see the password entered or not  
 """  
 # Get the current position and size of the existing text control* ctrls = {**"old"**: self.old\_password\_ctrl, **"new"**: self.new\_password\_ctrl, **"con"**: self.confirm\_password\_ctrl}  
 name = getattr(event.GetEventObject(), **'name'**, **None**)  
 open\_or\_closed = self.hidden\_bitmap **if not** self.is\_showing\_password **else** self.shown\_bitmap  
  
 **if** name **in** ctrls:  
 ctrl = ctrls[name]  
  
 position = ctrl.GetPosition()  
 size = ctrl.GetSize()  
  
 *# Create a new text control with the updated style* new\_style = wx.TE\_PASSWORD **if not** self.is\_showing\_password **else** wx.TE\_PROCESS\_ENTER  
 new\_textctrl = wx.TextCtrl(self.panel, pos=position, size=size, style=new\_style)  
  
 *# Copy the text from the existing text control to the new one* new\_textctrl.SetValue(ctrl.GetValue())  
  
 *# Update the reference to the new text control* **if** name == **"old"**:  
 self.old\_pass\_icon.SetBitmap(open\_or\_closed)  
 self.panel.GetSizer().Replace(self.old\_password\_ctrl, new\_textctrl)  
 self.old\_password\_ctrl.Destroy()  
 self.old\_password\_ctrl = new\_textctrl  
 **elif** name == **"new"**:  
 self.new\_pass\_icon.SetBitmap(open\_or\_closed)  
 self.panel.GetSizer().Replace(self.new\_password\_ctrl, new\_textctrl)  
 self.new\_password\_ctrl.Destroy()  
 self.new\_password\_ctrl = new\_textctrl  
 **else**:  
 self.con\_pass\_icon.SetBitmap(open\_or\_closed)  
 self.panel.GetSizer().Replace(self.confirm\_password\_ctrl, new\_textctrl)  
 self.confirm\_password\_ctrl.Destroy()  
 self.confirm\_password\_ctrl = new\_textctrl  
  
 self.is\_showing\_password = **not** self.is\_showing\_password  
  
 **def** GetValues(self):  
 *"""  
 :return: the values entered  
 """* **return** [self.old\_password\_ctrl.GetValue(), self.new\_password\_ctrl.GetValue(),  
 self.confirm\_password\_ctrl.GetValue()]  
  
 **def** on\_ok(self, event):  
 *"""  
 :param event: event on click  
 :return: ends the event with wx.ID\_OK id  
 """* self.EndModal(wx.ID\_OK)  
  
 **def** on\_cancel(self, event):  
 *"""  
 :param event: event on click  
 :return: ends the event with wx.ID\_CANCEL id  
 """* self.EndModal(wx.ID\_CANCEL)  
  
  
**class** ConfirmMailDialog(wx.Dialog):  
 **def** \_\_init\_\_(self, parent, title):  
 *"""  
 :param parent: panel parent  
 :param title: title of the dialog  
 """* super(ConfirmMailDialog, self).\_\_init\_\_(parent, title=title, size=(388, 150), pos=(960 - 174, 540 - 115))  
  
 self.panel = wx.Panel(self)  
  
 self.sizer = wx.BoxSizer(wx.VERTICAL)  
 self.sizer.AddSpacer(2)  
  
 label = wx.StaticText(self.panel, label=**"Enter the 6-digit code sent to your mail:"**)  
 self.sizer.Add(label, 0, wx.ALL, 5)  
  
 self.text\_ctrl = wx.TextCtrl(self.panel, size=(340, -1))  
 self.sizer.Add(self.text\_ctrl, 0, wx.CENTER)  
  
 self.remember\_checkbox = wx.CheckBox(self.panel, label=**"Don't ask again on this computer."**)  
 self.sizer.Add(self.remember\_checkbox, 0, wx.ALL, 5)  
  
 *# Create OK and Cancel buttons* self.ok\_button = wx.Button(self.panel, label=**"OK"**)  
 self.cancel\_button = wx.Button(self.panel, label=**"Cancel"**)  
  
 *# Bind events to buttons* self.ok\_button.Bind(wx.EVT\_BUTTON, self.on\_ok)  
 self.cancel\_button.Bind(wx.EVT\_BUTTON, self.on\_cancel)  
  
 buttons\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 buttons\_sizer.Add(self.ok\_button, 0, wx.ALL)  
 buttons\_sizer.AddSpacer(10)  
 buttons\_sizer.Add(self.cancel\_button, 0, wx.ALL)  
 buttons\_sizer.AddSpacer(15)  
  
 self.sizer.Add(buttons\_sizer, 0, wx.ALIGN\_RIGHT)  
  
 self.panel.SetSizerAndFit(self.sizer)  
  
 **def** GetValues(self):  
 *"""  
 :return: get the values entered  
 """* **return** [self.text\_ctrl.GetValue(), self.remember\_checkbox.GetValue()]  
  
 **def** on\_ok(self, event):  
 *"""  
 :param event: event on click  
 :return: ends the event with wx.ID\_OK id  
 """* self.EndModal(wx.ID\_OK)  
  
 **def** on\_cancel(self, event):  
 *"""  
 :param event: event on click  
 :return: ends the event with wx.ID\_CANCEL id  
 """* self.EndModal(wx.ID\_CANCEL)  
  
  
**class** ForgotPasswordDialog(wx.Dialog):  
 **def** \_\_init\_\_(self, parent, title):  
 *"""  
 :param parent: panel parent  
 :param title: title of the dialog  
 """* super(ForgotPasswordDialog, self).\_\_init\_\_(parent, title=title, size=(388, 185), pos=(960-174, 540-115))  
  
 self.panel = wx.Panel(self)  
  
 self.is\_showing\_password = **False** hidden\_path = **f"{**Settings.USER\_FILES\_PATH**}\\hidden-eye.png"** open\_path = **f"{**Settings.USER\_FILES\_PATH**}\\open-eye.png"** self.hidden\_bitmap = wx.Bitmap(hidden\_path, wx.BITMAP\_TYPE\_ANY)  
 self.shown\_bitmap = wx.Bitmap(open\_path, wx.BITMAP\_TYPE\_ANY)  
  
 *# Create two text boxes* self.new\_password\_ctrl = wx.TextCtrl(self.panel, size=(308, -1), style=wx.TE\_PASSWORD)  
 self.confirm\_password\_ctrl = wx.TextCtrl(self.panel, size=(308, -1), style=wx.TE\_PASSWORD)  
  
 *# Create OK and Cancel buttons* self.ok\_button = wx.Button(self.panel, label=**"OK"**)  
 self.cancel\_button = wx.Button(self.panel, label=**"Cancel"**)  
  
 *# Bind events to buttons* self.ok\_button.Bind(wx.EVT\_BUTTON, self.on\_ok)  
 self.cancel\_button.Bind(wx.EVT\_BUTTON, self.on\_cancel)  
  
 *# Set up the layout using a box sizer* sizer = wx.BoxSizer(wx.VERTICAL)  
 entries = wx.BoxSizer(wx.VERTICAL)  
  
 entries.Add(wx.StaticText(self.panel, label=**"Enter new password:"**), 0, wx.LEFT | wx.ALL)  
  
 new\_password\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 new\_password\_sizer.Add(self.new\_password\_ctrl, 0, wx.CENTER | wx.ALL, 5)  
 self.new\_pass\_icon = wx.StaticBitmap(self.panel, wx.ID\_ANY, self.hidden\_bitmap)  
 self.new\_pass\_icon.name = **"new"** new\_password\_sizer.Add(self.new\_pass\_icon)  
 entries.Add(new\_password\_sizer)  
  
 entries.AddSpacer(2)  
 entries.Add(wx.StaticText(self.panel, label=**"Confirm password:"**), 0, wx.LEFT | wx.ALL)  
  
 con\_password\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 con\_password\_sizer.Add(self.confirm\_password\_ctrl, 0, wx.CENTER | wx.ALL, 5)  
 self.con\_pass\_icon = wx.StaticBitmap(self.panel, wx.ID\_ANY, self.hidden\_bitmap)  
 self.con\_pass\_icon.name = **"con"** con\_password\_sizer.Add(self.con\_pass\_icon)  
 entries.Add(con\_password\_sizer)  
  
 entries.AddSpacer(5)  
 sizer.Add(entries, 0, wx.CENTER | wx.ALL)  
  
 buttons\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 buttons\_sizer.Add(self.ok\_button, 0, wx.ALL)  
 buttons\_sizer.AddSpacer(10)  
 buttons\_sizer.Add(self.cancel\_button, 0, wx.ALL)  
 buttons\_sizer.AddSpacer(15)  
  
 sizer.Add(buttons\_sizer, 0, wx.ALIGN\_RIGHT)  
  
 self.new\_pass\_icon.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
 self.con\_pass\_icon.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
  
 self.panel.SetSizerAndFit(sizer)  
  
 **def** change\_visibility(self, event):  
 *"""  
 :param event: event  
 :return: changes if you can see the password entered or not  
 """  
 # Get the current position and size of the existing text control* ctrls = {**"new"**: self.new\_password\_ctrl, **"con"**: self.confirm\_password\_ctrl}  
 name = getattr(event.GetEventObject(), **'name'**, **None**)  
 open\_or\_closed = self.hidden\_bitmap **if not** self.is\_showing\_password **else** self.shown\_bitmap  
  
 **if** name **in** ctrls:  
 ctrl = ctrls[name]  
  
 position = ctrl.GetPosition()  
 size = ctrl.GetSize()  
  
 *# Create a new text control with the updated style* new\_style = wx.TE\_PASSWORD **if not** self.is\_showing\_password **else** wx.TE\_PROCESS\_ENTER  
 new\_textctrl = wx.TextCtrl(self.panel, pos=position, size=size, style=new\_style)  
  
 *# Copy the text from the existing text control to the new one* new\_textctrl.SetValue(ctrl.GetValue())  
  
 *# Update the reference to the new text control* **if** name == **"new"**:  
 self.new\_pass\_icon.SetBitmap(open\_or\_closed)  
 self.panel.GetSizer().Replace(self.new\_password\_ctrl, new\_textctrl)  
 self.new\_password\_ctrl.Destroy()  
 self.new\_password\_ctrl = new\_textctrl  
 **else**:  
 self.con\_pass\_icon.SetBitmap(open\_or\_closed)  
 self.panel.GetSizer().Replace(self.confirm\_password\_ctrl, new\_textctrl)  
 self.confirm\_password\_ctrl.Destroy()  
 self.confirm\_password\_ctrl = new\_textctrl  
  
 self.is\_showing\_password = **not** self.is\_showing\_password  
  
 **def** GetValues(self):  
 *"""* **:return***: the values entered  
 """* **return** [self.new\_password\_ctrl.GetValue(), self.confirm\_password\_ctrl.GetValue()]  
  
 **def** on\_ok(self, event):  
 *"""* **:param** *event: event on click* **:return***: ends the event with wx.ID\_OK id  
 """* self.EndModal(wx.ID\_OK)  
  
 **def** on\_cancel(self, event):  
 *"""* **:param** *event: event on click* **:return***: ends the event with wx.ID\_CANCEL id  
 """* self.EndModal(wx.ID\_CANCEL)  
  
  
**class** TransparentText(wx.StaticText):  
 **def** \_\_init\_\_(self, parent, id=wx.ID\_ANY, label=**''**, pos=wx.DefaultPosition, size=wx.DefaultSize,  
 style=wx.TRANSPARENT\_WINDOW, name=**'transparenttext'**):  
 *"""  
 :param parent: panel parent  
 :param label: label of the text  
 :param style: style of the text  
 :param name: name of the text object  
 """* wx.StaticText.\_\_init\_\_(self, parent, id, label,  
 pos, size, style, name)  
  
 self.Bind(wx.EVT\_PAINT, self.on\_paint)  
 self.Bind(wx.EVT\_SIZE, self.on\_size)  
  
 **def** on\_paint(self, event):  
 *"""  
 :param event: event of on paint  
 :return: paints the text with transparent background  
 """* bdc = wx.PaintDC(self)  
 dc = wx.GCDC(bdc)  
  
 font\_face = self.GetFont()  
 font\_color = self.GetForegroundColour()  
  
 dc.SetFont(font\_face)  
 dc.SetTextForeground(font\_color)  
 dc.DrawText(self.GetLabel(), 0, 0)  
  
 **def** on\_size(self, event):  
 *"""  
 :param event: event of on size  
 :return: refreshes when screen changes size  
 """* self.Refresh()  
 event.Skip()

filesPanel:  
**import** wx  
**import** wx.lib.scrolledpanel  
**import** io  
**import** os  
**from** pubsub **import** pub  
**import** time  
**from** .userPanel **import** UserPanel  
**from** .customMenusAndDialogs **import** UserMenuFeatures, FileMenuFeatures, FolderMenuFeatures, TransparentText  
**from** settings **import** CurrentSettings **as** Settings  
**from** ClientFiles **import** clientProtocol  
  
  
**class** MyDropTarget(wx.DropTarget):  
 **def** \_\_init\_\_(self, panel):  
 *"""* **:param** *panel: parent panel  
 """* wx.DropTarget.\_\_init\_\_(self)  
 self.panel = panel  
  
 self.name = wx.TextDataObject()  
 self.SetDataObject(self.name)  
  
 self.data = wx.TextDataObject()  
 self.SetDataObject(self.data)  
  
 **def** OnDragOver(self, x, y, default):  
 *"""* **:param** *x: x position* **:param** *y: y position* **:param** *default: default cursor to show* **:return***: which cursor to show depending on where the mouse is  
 """  
 # Check if the cursor is above an image using the panel's method* cursor\_to\_show = wx.DragNone  
 isAbove, name = self.panel.is\_cursor\_above\_image()  
  
 **if** isAbove:  
 *# Visual feedback for a valid drop location* cursor\_to\_show = wx.DragCopy  
 self.data = name  
  
 **return** cursor\_to\_show  
  
 **def** OnData(self, x, y, result):  
 *"""* **:param** *x: x position* **:param** *y: y position* **:param** *result: is the drop ok* **:return***: result  
 """  
 # Get the dropped data* dragged\_name = self.name.GetText()  
 dropped\_name = self.data.GetText()  
 *# Process the dropped data* **if** dragged\_name **and** dropped\_name:  
 self.panel.handle\_dropped\_item(dragged\_name, dropped\_name)  
  
 **return** result  
  
  
**class** FilesPanel(wx.Panel):  
 **def** \_\_init\_\_(self, parent, frame, comm):  
 *"""* **:param** *parent: panel parent* **:param** *frame: frame parent* **:param** *comm: comm object of user  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(1920, 1080), style=wx.SIMPLE\_BORDER)  
  
 self.grid\_sizer = **None** self.frame = frame  
 self.comm = comm  
 self.parent = parent  
  
 self.sizer = wx.BoxSizer(wx.VERTICAL)  
  
 button\_font = wx.Font(15, wx.DEFAULT, wx.NORMAL, wx.BOLD)  
  
 self.title\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 self.title = TransparentText(self, -1)  
 titlefont = wx.Font(60, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, 0, **"High tower text"**)  
 self.title.SetFont(titlefont)  
  
 self.title\_sizer.Add(self.title)  
  
 self.curPath = **None** self.pathBeforeSharedFiles = **None** self.copied\_file = **None** self.progressDialog = **None** self.file\_name = **""** self.filesObj = {}  
  
 self.scroll\_panel = wx.lib.scrolledpanel.ScrolledPanel(self, -1, size=(1670, 800), style=wx.SIMPLE\_BORDER)  
 self.scroll\_panel.SetupScrolling()  
 self.scroll\_panel.SetBackgroundColour(**"white"**)  
  
 self.icons\_and\_files\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 self.icons\_sizer = wx.BoxSizer(wx.VERTICAL)  
  
 self.settings\_img = wx.BitmapButton(self, wx.ID\_ANY, wx.Bitmap(**f"{**Settings.USER\_FILES\_PATH**}/Settings.png"**,  
 wx.BITMAP\_TYPE\_ANY))  
 tooltip = wx.ToolTip(**"Settings"**)  
 self.settings\_img.SetToolTip(tooltip)  
  
 self.shared\_files\_img = wx.BitmapButton(self, wx.ID\_ANY, wx.Bitmap(**f"{**Settings.USER\_FILES\_PATH**}/Shared.png"**,  
 wx.BITMAP\_TYPE\_ANY))  
 tooltip = wx.ToolTip(**"Shared files"**)  
 self.shared\_files\_img.SetToolTip(tooltip)  
  
 self.shared\_files\_img.Bind(wx.EVT\_BUTTON, self.show\_shared\_files)  
 self.settings\_img.Bind(wx.EVT\_BUTTON, self.show\_settings)  
  
 self.icons\_sizer.AddSpacer(10)  
 self.icons\_sizer.Add(self.settings\_img)  
 self.icons\_sizer.AddSpacer(22)  
 self.icons\_sizer.Add(self.shared\_files\_img)  
  
 self.files\_sizer = wx.BoxSizer()  
 self.icons\_and\_files\_sizer.AddSpacer(90)  
 self.icons\_and\_files\_sizer.Add(self.scroll\_panel, 0, wx.CENTER)  
 self.icons\_and\_files\_sizer.AddSpacer(17)  
 self.icons\_and\_files\_sizer.Add(self.icons\_sizer)  
  
 self.scroll\_panel.SetSizer(self.files\_sizer)  
 self.scroll\_panel.Bind(wx.EVT\_RIGHT\_DOWN, self.show\_user\_options\_menu)  
  
 self.buttons\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 self.login\_button = wx.Button(self, label=**"BACK TO LOGIN"**)  
 self.login\_button.SetFont(button\_font)  
 self.backButton = wx.Button(self, label=**"BACK"**)  
 self.backButton.SetFont(button\_font)  
 self.uploadButton = wx.Button(self, label=**"UPLOAD FILE"**)  
 self.uploadButton.SetFont(button\_font)  
 self.createDirButton = wx.Button(self, label=**"CREATE DIRECTORY"**)  
 self.createDirButton.SetFont(button\_font)  
 self.pasteButton = wx.Button(self, label=**"PASTE"**)  
 self.pasteButton.SetFont(button\_font)  
  
 self.login\_button.Bind(wx.EVT\_BUTTON, self.login\_control)  
 self.backButton.Bind(wx.EVT\_BUTTON, self.back\_dir)  
 self.uploadButton.Bind(wx.EVT\_BUTTON, self.upload\_file\_request)  
 self.createDirButton.Bind(wx.EVT\_BUTTON, self.create\_dir\_request)  
 self.pasteButton.Bind(wx.EVT\_BUTTON, self.paste\_file\_request)  
  
 self.buttons\_sizer.Add(self.backButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.login\_button)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.uploadButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.createDirButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.pasteButton)  
  
 self.sizer.AddMany([(self.title\_sizer, 0, wx.CENTER),  
 (self.icons\_and\_files\_sizer, 0, wx.CENTER)])  
  
 self.sizer.AddSpacer(15)  
 self.sizer.Add(self.buttons\_sizer, 0, wx.CENTER)  
  
 self.SetSizer(self.sizer)  
  
 pub.subscribe(self.\_get\_branches, **"filesOk"**)  
 pub.subscribe(self.\_delete\_obj, **"deleteOk"**)  
 pub.subscribe(self.\_rename\_obj, **"renameOk"**)  
 pub.subscribe(self.\_upload\_object, **"uploadOk"**)  
 pub.subscribe(self.\_create\_dir, **"createOk"**)  
 pub.subscribe(self.\_add\_shared\_file, **"addFile"**)  
 pub.subscribe(self.\_move\_file, **"moveFile"**)  
 pub.subscribe(self.\_handle\_paste\_file, **"pasteFile"**)  
 pub.subscribe(self.\_show\_progress\_bar, **"startBar"**)  
 pub.subscribe(self.\_change\_progress\_bar, **"changeProgress"**)  
 pub.subscribe(self.change\_settings\_to\_profile, **"changeSettingsToPhoto"**)  
 pub.subscribe(self.\_upload\_object, **"zipFileOk"**)  
  
 self.Layout()  
  
 self.drop\_target = MyDropTarget(self)  
 self.SetDropTarget(self.drop\_target)  
 self.Bind(wx.EVT\_PAINT, self.PaintBackgroundImage)  
  
 self.Hide()  
  
 **def** PaintBackgroundImage(self, evt):  
 *"""* **:param** *evt: on paint event* **:return***: showing the background image  
 """* dc = wx.PaintDC(self)  
  
 bmp = wx.Bitmap(**rf"{**Settings.USER\_FILES\_PATH**}/bg.png"**)  
 dc.DrawBitmap(bmp, 0, 0)  
  
 **def** change\_settings\_to\_profile(self):  
 *"""* **:return***: changes the photo from the settings to the profile photo  
 """* image = wx.Image(io.BytesIO(self.parent.profilePhoto), wx.BITMAP\_TYPE\_ANY)  
 image.Rescale(108, 108)  
  
 *# Convert the wx.Image to a wx.Bitmap* bitmap = wx.Bitmap(image)  
  
 self.settings\_img.SetBitmap(bitmap)  
  
 **def** show\_user\_options\_menu(self, event):  
 *"""* **:param** *event: on click event* **:return***: showing user menu  
 """* flag = **True  
 if** self.curPath == **""**:  
 flag = **False** self.PopupMenu(UserMenuFeatures(self, flag))  
  
 **def** start\_drag(self, event):  
 *"""* **:param** *event: on drag event* **:return***: starts the drag event  
 """  
 # Start a drag-and-drop operation when an item is left-clicked* item\_text = event.GetEventObject()  
 data = wx.TextDataObject(item\_text.GetName())  
 self.drop\_target.name = data  
 drop\_source = wx.DropSource(item\_text)  
 drop\_source.SetData(data)  
 drop\_source.DoDragDrop(wx.Drag\_CopyOnly)  
  
 **def** handle\_dropped\_item(self, item\_dropped, dropped\_to\_item):  
 *"""* **:param** *item\_dropped: name of the item dropped* **:param** *dropped\_to\_item: name of the item dropped onto* **:return***: requests to move the file a folder from the server  
 """  
 # Process the dropped item* self.file\_name = item\_dropped  
 src = **f"{**self.parent.username**}/{**self.curPath**}/{**item\_dropped**}"**.replace(**'//'**, **'/'**)  
 dst = **f"{**self.parent.username**}/{**self.curPath**}/{**dropped\_to\_item**}"**.replace(**'//'**, **'/'**)  
  
 msg = clientProtocol.pack\_move\_file\_folder\_request(src, dst)  
 self.comm.send(msg)  
  
 **def** is\_cursor\_above\_image(self):  
 *"""* **:return***: is the cursor above a directory image  
 """  
 # Get the mouse position* mouse\_position = wx.GetMousePosition()  
 screen\_position = self.scroll\_panel.GetScreenPosition()  
 mouse\_x, mouse\_y = mouse\_position.x - screen\_position.x, mouse\_position.y - screen\_position.y  
 flag = **False** text\_data\_object = **None** *# Iterate over sizer items to check if the cursor is over an image* **for** item **in** self.scroll\_panel.GetChildren():  
 item\_rect = item.GetRect()  
 **if** self.filesObj[**f"{**self.curPath**}/{**item.GetName()**}"**.lstrip(**'/'**)][1]:  
 **if** item\_rect.Contains(mouse\_x, mouse\_y):  
 flag = **True** text\_data\_object = wx.TextDataObject()  
 text\_data\_object.SetText(item.GetName())  
  
 **return** flag, text\_data\_object  
  
 **def** login\_control(self, event):  
 *"""* **:param** *event: on click event* **:return***: changes back to the login screen  
 """* self.parent.change\_screen(self, self.parent.login)  
  
 **def** \_get\_branches(self, branches):  
 *"""* **:param** *branches: users branches from the server* **:return***: updates the branches and shows them  
 """* self.branches = branches  
 self.copied\_file = **None  
  
 for** branch **in** branches:  
 branch[1].sort()  
 branch[2].sort()  
 **if** branch[1] == [**""**]:  
 branch[1].pop(0)  
 **if** branch[2] == [**""**]:  
 branch[2].pop(0)  
  
 **for** branch **in** branches:  
 **if** branch[0] == **""**:  
 branch[1].remove(**"@#$SHAREDFILES$#@"**)  
 self.curPath = **""** self.show\_files(self.curPath)  
  
 **def** show\_files(self, path):  
 *"""* **:param** *path: path of branch to show* **:return***: shows the correct files on the screen  
 """* self.scroll\_panel.DestroyChildren()  
  
 **if "@#$SHAREDFILES$#@" not in** self.curPath.split(**'/'**):  
 self.shared\_files\_img.Show()  
 **else**:  
 split\_path = self.curPath.split(**'/'**)  
 path = **'/'**.join(split\_path[split\_path.index(**"@#$SHAREDFILES$#@"**):])  
  
 **if** path == **""**:  
 self.backButton.Show(**False**)  
 **else**:  
 self.backButton.Show(**True**)  
  
 branch = [**""**, [], []]  
 **for** temp **in** self.branches:  
 **if** temp[0] == path:  
 branch = temp  
  
 dirs = branch[1][::]  
 files = branch[2][::]  
  
 self.grid\_sizer = wx.GridSizer(cols=15, hgap=10, vgap=10)  
  
 image\_types = [**"apng"**, **"avif"**, **"gif"**, **"jpg"**, **"jpeg"**, **"jfif"**, **"pjpeg"**, **"pjp"**, **"png"**, **"svg"**, **"webp"**, **"bmp"**, **"ico"**,  
 **"cur"**, **"tif"**, **"tiff"**]  
 file\_image\_paths = {**'txt'**: **f"{**Settings.USER\_FILES\_PATH**}/text\_icon.png"**,  
 \*\*{img: **f"{**Settings.USER\_FILES\_PATH**}/photo\_icon.png" for** img **in** image\_types},  
 **'zip'**: **f"{**Settings.USER\_FILES\_PATH**}/zip\_icon.png"**,  
 **'docx'**: **f"{**Settings.USER\_FILES\_PATH**}/docx\_icon.png"**,  
 **'doc'**: **f"{**Settings.USER\_FILES\_PATH**}/docx\_icon.png"**,  
 **'pptx'**: **f"{**Settings.USER\_FILES\_PATH**}/pptx\_icon.png"**,  
 **'ppt'**: **f"{**Settings.USER\_FILES\_PATH**}/pptx\_icon.png"**,  
 **'pdf'**: **f"{**Settings.USER\_FILES\_PATH**}/pdf\_icon.png"**,  
 **'xls'**: **f"{**Settings.USER\_FILES\_PATH**}/xlsx\_icon.png"**,  
 **'xlsx'**: **f"{**Settings.USER\_FILES\_PATH**}/xlsx\_icon.png"**,  
 **'default'**: **f"{**Settings.USER\_FILES\_PATH**}/default\_icon.png"**}  
  
 directory\_image\_path = **f"{**Settings.USER\_FILES\_PATH**}/dirs\_icon.png"** *# Add items with corresponding images to the grid sizer* **while** dirs **or** files:  
 dir\_file\_flag = **False  
 if** dirs:  
 item = dirs.pop(0)  
 image\_path = directory\_image\_path  
 dir\_file\_flag = **True  
 elif** files:  
 item = files.pop(0)  
 file\_type = item[item.rfind(**'.'**)+1:]  
  
 **if** file\_type **in** file\_image\_paths:  
 image\_path = file\_image\_paths[file\_type]  
 **else**:  
 image\_path = file\_image\_paths[**'default'**]  
  
 **else**:  
 **continue** item\_sizer = wx.BoxSizer(wx.VERTICAL)  
  
 *# Add image* item\_image = wx.StaticBitmap(self.scroll\_panel, -1, wx.Bitmap(image\_path, wx.BITMAP\_TYPE\_ANY),  
 size=(100, 80), name=item)  
  
 item\_sizer.Add(item\_image, 0, wx.ALL)  
  
 *# Add item text* item\_text = wx.StaticText(self.scroll\_panel, label=item[:10] + **"..." if** len(item) > 13 **else** item, name=item)  
 item\_sizer.Add(item\_text, 0, wx.CENTER)  
  
 **if** dir\_file\_flag:  
 item\_text.Bind(wx.EVT\_LEFT\_DOWN, self.select\_file)  
 item\_image.Bind(wx.EVT\_LEFT\_DOWN, self.select\_file)  
 **else**:  
 item\_text.Bind(wx.EVT\_LEFT\_DOWN, self.start\_drag)  
 item\_image.Bind(wx.EVT\_LEFT\_DOWN, self.start\_drag)  
  
 item\_text.Bind(wx.EVT\_RIGHT\_DOWN, self.select\_file)  
 item\_image.Bind(wx.EVT\_RIGHT\_DOWN, self.select\_file)  
  
 self.filesObj[**f"{**path**}/{**item**}"**.lstrip(**'/'**)] = (item\_sizer, dir\_file\_flag)  
  
 *# Add item sizer to grid\_sizer* self.grid\_sizer.Add(item\_sizer, 0, wx.CENTER)  
  
 *# Set the grid sizer for the scrollable panel* self.files\_sizer.Add(self.grid\_sizer, 0, wx.TOP)  
  
 *# Scroll to the top* self.scroll\_panel.Scroll(0, 0)  
  
 *# Refresh the layout* self.sizer.Layout()  
  
 **def** select\_file(self, event):  
 *"""* **:param** *event: on click event* **:return***: shows the correct menu if its a file or a folder  
 """* obj = event.GetEventObject()  
  
 path\_to\_find = **f"{**self.curPath**}/{**obj.GetName()**}"**.lstrip(**'/'**)  
  
 **if** path\_to\_find **not in** self.filesObj **and "@#$SHAREDFILES$#@" in** path\_to\_find:  
 split\_path = self.curPath.split(**'/'**)  
 path\_to\_find = **'/'**.join(split\_path[split\_path.index(**"@#$SHAREDFILES$#@"**):]) + **'/'** + obj.GetName()  
  
 *# if left clicked a directory* **if** self.filesObj[path\_to\_find][1] **and** event.GetEventType() == wx.EVT\_LEFT\_DOWN.typeId:  
 self.chose\_dir(obj.GetName())  
  
 *# if right clicked a directory* **elif** self.filesObj[path\_to\_find][1]:  
 self.PopupMenu(FolderMenuFeatures(self, name=obj.GetName()))  
  
 *# if right clicked a file* **else**:  
 self.PopupMenu(FileMenuFeatures(self, name=obj.GetName()))  
  
 **def** chose\_dir(self, name):  
 *"""* **:param** *name: name of the directory chosen* **:return***: showing the files inside the directory chosen  
 """* self.curPath += **f"/{**name**}"** self.curPath = self.curPath.lstrip(**'/'**)  
 self.show\_files(self.curPath)  
  
 **def** back\_dir(self, event):  
 *"""* **:param** *event: on click event* **:return***: going back a directory and showing the files  
 """* **if** self.curPath == **"@#$SHAREDFILES$#@"**:  
 self.curPath = self.pathBeforeSharedFiles  
 **else**:  
 self.curPath = **"/"**.join(self.curPath.split(**'/'**)[:-1])  
  
 self.show\_files(self.curPath)  
  
 **def** delete\_file\_request(self, name):  
 *"""* **:param** *name: name of the item to delete* **:return***: sends item delete request  
 """* dlg = wx.MessageDialog(self, **'Do you want to delete?'**, **'Confirmation'**, wx.YES\_NO | wx.ICON\_QUESTION)  
 result = dlg.ShowModal()  
 dlg.Destroy()  
  
 **if** result == wx.ID\_YES:  
 self.file\_name = name  
 full\_path = **f"{**self.parent.username**}/{**self.curPath**}/{**self.file\_name**}"**.lstrip(**"/"**)  
 msg2send = clientProtocol.pack\_delete\_request(full\_path)  
 self.comm.send(msg2send)  
  
 **def** \_delete\_obj(self):  
 *"""* **:return***: deletes the item from the branch  
 """* is\_dir = self.filesObj[**f"{**self.curPath**}/{**self.file\_name**}"**.lstrip(**'/'**)][1]  
 branches\_to\_remove = []  
  
 **for** branch **in** self.branches:  
 **if** branch[0] == self.curPath:  
 **if** is\_dir:  
 branch[1].remove(self.file\_name)  
 **else**:  
 branch[2].remove(self.file\_name)  
  
 **elif** branch[0].startswith(**f"{**self.curPath**}/{**self.file\_name**}/"**.lstrip(**'/'**)) **or** branch[0] == self.file\_name\  
 **and** is\_dir:  
 branches\_to\_remove.append(branch)  
  
 **for** branch\_to\_remove **in** branches\_to\_remove:  
 self.branches.remove(branch\_to\_remove)  
  
 self.show\_files(self.curPath)  
 self.parent.show\_pop\_up(**f"Deleted {**self.file\_name**} successfully."**, **"Success"**)  
  
 **def** rename\_file\_request(self, name):  
 *"""* **:param** *name: file to rename* **:return***: send rename file request  
 """* dlg = wx.TextEntryDialog(self, **f'Do you want to rename {**name**}?'**, **'Confirmation'**, **''**)  
 result = dlg.ShowModal()  
 dlg.Destroy()  
  
 **if** result == wx.ID\_OK:  
 self.file\_name = name  
 full\_path = **f"{**self.parent.username**}/{**self.curPath**}/{**name**}"**.lstrip(**"/"**)  
 msg2send = clientProtocol.pack\_rename\_file\_request(full\_path, dlg.GetValue())  
 self.comm.send(msg2send)  
  
 **def** \_rename\_obj(self, new\_name):  
 *"""  
 :param new\_name: new name of file to rename  
 :return: renames item  
 """* is\_dir = self.filesObj[**f"{**self.curPath**}/{**self.file\_name**}"**.lstrip(**'/'**)][1]  
  
 **for** branch **in** self.branches:  
 **if** branch[0] == self.curPath:  
 **if** is\_dir:  
 index\_to\_replace = branch[1].index(self.file\_name)  
 branch[1][index\_to\_replace] = new\_name  
 **else**:  
 index\_to\_replace = branch[2].index(self.file\_name)  
 branch[2][index\_to\_replace] = new\_name  
 **break  
  
 for** index **in** range(len(self.branches)):  
 **if not** is\_dir:  
 **break** names\_of\_path = self.branches[index][0].split(**'/'**)  
 **if** self.branches[index][0].startswith(**f"{**self.curPath**}/{**self.file\_name**}/"**.lstrip(**'/'**)) **or** \  
 self.branches[index][0] == self.file\_name:  
  
 index\_of\_name = names\_of\_path.index(self.file\_name)  
 names\_of\_path[index\_of\_name] = new\_name  
 branch = list(self.branches[index])  
 name = **'/'**.join(names\_of\_path)  
 branch[0] = name  
  
 self.branches[index] = tuple(branch)  
  
 self.show\_files(self.curPath)  
 self.parent.show\_pop\_up(**f"Renamed {**self.file\_name**} to {**new\_name**} successfully."**, **"Success"**)  
  
 **def** download\_file\_request(self, file\_name):  
 *"""  
 :param file\_name: name of file to download  
 :return: sends download file request  
 """* base\_name, file\_extension = os.path.splitext(file\_name)  
  
 file\_type = file\_extension[1:].lower()  
  
 type\_descriptions = {  
 **'txt'**: **'Text files'**,  
 **'docx'**: **'Microsoft Word documents'**,  
 **'jpg'**: **'JPEG images'**,  
 **'png'**: **'PNG images'**,  
 }  
  
 *# creating wild card for download type* wildcard\_list = [**f"{**type\_descriptions[key]**} (\*.{**key.lower()**})|\*.{**key.lower()**}" for** key **in** type\_descriptions]  
 **if** file\_type **in** type\_descriptions:  
 wildcard\_list.remove(**f"{**type\_descriptions[file\_type]**} (\*.{**file\_type**})|\*.{**file\_type**}"**)  
 wildcard\_list.insert(0, **f"{**type\_descriptions[file\_type]**} (\*.{**file\_type**})|\*.{**file\_type**}"**)  
 wildcard\_list.append(**"All files (\*.\*)|\*.\*"**)  
 **else**:  
 wildcard\_list.insert(0, **"All files (\*.\*)|\*.\*"**)  
  
 wildcard = **'|'**.join(wildcard\_list)  
  
 dlg = wx.FileDialog(self, **"Choose a file"**, defaultFile=file\_name, wildcard=wildcard, style=wx.FD\_SAVE)  
 result = dlg.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 selected\_path = dlg.GetPath().replace(**'\\'**, **'/'**)  
  
 msg2send = clientProtocol.pack\_download\_file\_request(**f"{**self.parent.username**}/{**self.curPath**}/{**file\_name**}"**,  
 selected\_path)  
 self.parent.files\_comm.send(msg2send)  
  
 dlg.Destroy()  
  
 **def** upload\_file\_request(self, event):  
 *"""  
 :param event: on click event  
 :return: send upload file request  
 """* dlg = wx.FileDialog(self, **"Choose a file"**, style=wx.DD\_DEFAULT\_STYLE)  
 result = dlg.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 selected\_path = dlg.GetPath().replace(**'\\'**, **'/'**)  
 self.parent.files\_comm.send\_file(12, selected\_path, **f"{**self.parent.username**}/{**self.curPath**}"**.rstrip(**'/'**))  
  
 **def** \_upload\_object(self, path):  
 *"""  
 :param path: path of file in the server  
 :return: adds the file to the branches  
 """* name\_to\_add = path.split(**'/'**)[-1]  
 text\_to\_show = **f"Uploaded {**name\_to\_add**} successfully."  
 for** branch **in** self.branches:  
 **if** branch[0] == self.curPath:  
 **if** name\_to\_add **not in** branch[2]:  
 branch[2].append(name\_to\_add)  
 branch[2].sort()  
 **else**:  
 text\_to\_show = **f"Updated {**name\_to\_add**} successfully."  
 break** self.show\_files(self.curPath)  
 self.parent.show\_pop\_up(text\_to\_show, **"Success"**)  
  
 **def** create\_dir\_request(self, event):  
 *"""  
 :param event: on click event  
 :return: send create dir request  
 """* dlg = wx.TextEntryDialog(self, **f'Please enter the name for the directory:'**, **'Create new directory'**, **''**)  
 result = dlg.ShowModal()  
 dlg.Destroy()  
  
 **if** result == wx.ID\_OK:  
 **if** dlg.GetValue() != **"@#$SHAREDFILES$#@"**:  
 self.file\_name = dlg.GetValue()  
 full\_path = **f"{**self.parent.username**}/{**self.curPath**}/{**self.file\_name**}"**.lstrip(**"/"**)  
 msg2send = clientProtocol.pack\_create\_folder\_request(full\_path)  
 self.comm.send(msg2send)  
 **else**:  
 self.parent.show\_pop\_up(**f"Can't create a folder with that name, please choose a different name."**,  
 **"Error"**)  
  
 **def** \_create\_dir(self):  
 *"""  
 :return: adding the directory to the path  
 """* **for** branch **in** self.branches:  
 **if** branch[0] == self.curPath:  
 branch[1].append(self.file\_name)  
 branch[1].sort()  
 **break** self.branches.append((**f'{**self.curPath**}/{**self.file\_name**}'**.lstrip(**'/'**), [], []))  
 self.show\_files(self.curPath)  
  
 **def** share\_file\_request(self, name):  
 *"""  
 :param name: file name to share  
 :return: send share file request  
 """* dlg = wx.TextEntryDialog(self, **f'Enter username of user you want to share to:'**, **'Confirmation'**, **''**)  
 result = dlg.ShowModal()  
 dlg.Destroy()  
  
 **if** result == wx.ID\_OK:  
 username = dlg.GetValue()  
 **if** username != self.parent.username:  
 full\_path = **f"{**self.parent.username**}/{**self.curPath**}/{**name**}"**.replace(**'//'**, **'/'**)  
 msg = clientProtocol.pack\_share\_request(full\_path, username)  
 self.comm.send(msg)  
  
 **else**:  
 self.parent.show\_pop\_up(**f"Don't enter your own username."**, **"Error"**)  
  
 **def** show\_shared\_files(self, event):  
 *"""  
 :param event: event on click  
 :return: shows shared files  
 """* self.shared\_files\_img.Hide()  
 self.pathBeforeSharedFiles = self.curPath  
 self.curPath = **"@#$SHAREDFILES$#@"** self.show\_files(**"@#$SHAREDFILES$#@"**)  
  
 **def** \_add\_shared\_file(self, path):  
 *"""  
 :param path: path to add  
 :return: adds the file to the users shared files  
 """* user\_who\_shared, file = path.split(**'/'**)[0], path.split(**'/'**)[-1]  
 **for** branch **in** self.branches:  
 **if** branch[0] == **"@#$SHAREDFILES$#@"**:  
 **if** user\_who\_shared **not in** branch[1]:  
 branch[1].append(user\_who\_shared)  
 branch[1].sort()  
 **break** *# if the user who shared hasnent shared a file to this user before* **if not** any(branch[0] == **f"@#$SHAREDFILES$#@/{**user\_who\_shared**}" for** branch **in** self.branches):  
 self.branches.append((**f'@#$SHAREDFILES$#@/{**user\_who\_shared**}'**, [], []))  
 self.branches[-1][2].append(file)  
 self.branches[-1][2].sort()  
  
 *# if the user has shared before add it to his branch* **else**:  
 **for** branch **in** self.branches:  
 **if** branch[0] == **f"@#$SHAREDFILES$#@/{**user\_who\_shared**}"**:  
 print(file)  
 branch[2].append(file)  
 branch[2].sort()  
  
 self.show\_files(self.curPath)  
  
 **def** \_move\_file(self, path):  
 *"""  
 :param path: path to move the file from  
 :return: moves the file from the path to where is needed  
 """* path = **"/"**.join(path.split(**'/'**)[1::])  
  
 **for** branch **in** self.branches:  
 *# removing the file from the old location* **if** branch[0] == self.curPath:  
 branch[2].remove(self.file\_name)  
  
 *# adding the file to the new location* **if** branch[0] == path:  
 branch[2].append(self.file\_name)  
  
 self.show\_files(self.curPath)  
 self.parent.show\_pop\_up(**"Moved file successfully"**, **"Success"**)  
  
 **def** paste\_file\_request(self, event):  
 *"""  
 :param event: on click event  
 :return: sends paste file request  
 """* **if** self.copied\_file:  
 src = **f"{**self.parent.username**}/{**self.copied\_file**}"** dst = **f"{**self.parent.username**}/{**self.curPath**}"**.rstrip(**'/'**)  
  
 msg = clientProtocol.pack\_paste\_file\_request(src, dst)  
 self.comm.send(msg)  
 **else**:  
 self.parent.show\_pop\_up(**f"Your clipboard is empty."**, **"Error"**)  
  
 **def** \_handle\_paste\_file(self):  
 *"""  
 :return: adds the file pasted to the users files  
 """* file\_name = self.copied\_file.split(**'/'**)[-1]  
 text\_to\_show = **"Pasted file successfully."  
  
 for** branch **in** self.branches:  
 **if** branch[0] == self.curPath:  
 **if** file\_name **not in** branch[2]:  
 branch[2].append(file\_name)  
 branch[2].sort()  
 **else**:  
 text\_to\_show = **"Updated file successfully."** self.show\_files(self.curPath)  
 self.parent.show\_pop\_up(text\_to\_show, **"Success"**)  
  
 **def** open\_file\_request(self, name):  
 *"""  
 :param name: file name  
 :return: sends file edit/open request  
 """* msg2send = clientProtocol.pack\_open\_file\_request(  
 **f"{**self.parent.username**}/{**self.curPath**}/{**name**}"**.replace(**"//"**, **'/'**)  
 )  
 self.parent.files\_comm.send(msg2send)  
  
 **def** \_show\_progress\_bar(self, name, opcode):  
 *"""  
 :param name: name of file  
 :param opcode: opcode of message  
 :return: open the progress bar  
 """* msg = **"Download" if** opcode != **"04" else "Upload"** + **" at 0%"** self.progressDialog = wx.ProgressDialog(title=name, message=msg, maximum=100,  
 style=wx.PD\_APP\_MODAL | wx.PD\_AUTO\_HIDE)  
  
 **def** \_change\_progress\_bar(self, percent, opcode):  
 *"""  
 :param percent: percent to show  
 :param opcode: opcode of message  
 :return: changes the progress bar  
 """* **while not** self.progressDialog:  
 **pass** self.progressDialog.Update(percent, **f"Download at {**percent**}%" if** opcode != **"04" else f"Upload at {**percent**}%"**)  
 time.sleep(0.00001)  
  
 **if** percent == 100:  
 self.progressDialog.Destroy()  
  
 **def** show\_settings(self, event):  
 *"""  
 :param event: on click  
 :return: shows settings  
 """* settings = UserPanel(self.parent, self.frame, self.comm, self.parent.files\_comm)  
  
 self.parent.change\_screen(self, settings)  
  
 **def** zip\_folder\_request(self, name):  
 *"""  
 :param name: file name  
 :return: sends zip folder request  
 """* file\_path = **f"{**self.curPath**}/{**name**}"**.lstrip(**'/'**)  
 full\_path = **f"{**self.parent.username**}/{**file\_path**}"** msg = clientProtocol.pack\_zip\_folder\_request(full\_path)  
 self.comm.send(msg)

loginPanel:  
**import** wx  
**import** wx.adv  
**from** pubsub **import** pub  
**from** settings **import** CurrentSettings **as** Settings  
**from** ClientFiles **import** clientProtocol  
**from** .customMenusAndDialogs **import** ConfirmMailDialog, ForgotPasswordDialog, TransparentText  
  
  
**class** LoginPanel(wx.Panel):  
 **def** \_\_init\_\_(self, parent, frame, comm):  
 *"""* **:param** *parent: panel parent* **:param** *frame: frame parent* **:param** *comm: client comm object  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(1920, 1080), style=wx.TRANSPARENT\_WINDOW)  
 self.frame = frame  
 self.comm = comm  
 self.parent = parent  
  
 self.username\_input = **None** self.SetBackgroundColour(wx.WHITE)  
  
 image = wx.Image(**f"{**Settings.USER\_FILES\_PATH**}/info.png"**, wx.BITMAP\_TYPE\_ANY)  
  
 bitmap = wx.Bitmap(image)  
  
 info\_button = wx.StaticBitmap(self, wx.ID\_ANY, bitmap)  
 info\_button.SetTransparent(0)  
  
 title\_font = wx.Font(68, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**, **"High Tower Text"**)  
 text\_font = wx.Font(30, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**)  
 forgot\_password\_font = wx.Font(15, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**)  
 self.entry\_font = wx.Font(20, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**)  
 button\_font = wx.Font(15, wx.DEFAULT, wx.NORMAL, wx.BOLD)  
  
 hidden\_path = **f"{**Settings.USER\_FILES\_PATH**}\\hidden-eye.png"** open\_path = **f"{**Settings.USER\_FILES\_PATH**}\\open-eye.png"** self.is\_showing\_password = **False** self.hidden\_bitmap = wx.Bitmap(hidden\_path, wx.BITMAP\_TYPE\_ANY)  
 self.shown\_bitmap = wx.Bitmap(open\_path, wx.BITMAP\_TYPE\_ANY)  
  
 self.sizer = wx.BoxSizer(wx.VERTICAL)  
 entries = wx.BoxSizer(wx.VERTICAL)  
 forgot\_password\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 self.sizer.AddSpacer(290)  
 title = TransparentText(self, label=**"LOG IN"**, style=wx.TRANSPARENT\_WINDOW)  
 title.SetForegroundColour(wx.BLACK)  
 title.SetFont(title\_font)  
  
 name\_sizer = wx.BoxSizer(wx.VERTICAL)  
 name\_text = TransparentText(self, label=**"Username "**, style=wx.TRANSPARENT\_WINDOW)  
 name\_text.SetFont(text\_font)  
 self.nameField = wx.TextCtrl(self, name=**"username"**, size=(650, 40))  
 self.nameField.SetFont(self.entry\_font)  
  
 name\_sizer.Add(name\_text, 0, wx.Center, 5)  
 name\_sizer.Add(self.nameField)  
 name\_sizer.AddSpacer(15)  
  
 self.pass\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 pass\_text = TransparentText(self, label=**"Password "**, style=wx.TRANSPARENT\_WINDOW)  
 pass\_text.SetFont(text\_font)  
 self.passField = wx.TextCtrl(self, -1, name=**"password"**, size=(650, 40),  
 style=wx.TE\_PASSWORD | wx.TE\_PROCESS\_ENTER)  
 self.passField.Bind(wx.EVT\_TEXT\_ENTER, self.on\_ok)  
 self.passField.SetFont(self.entry\_font)  
  
 name\_sizer.Add(pass\_text)  
 self.pass\_sizer.AddSpacer(27)  
 self.pass\_sizer.Add(self.passField)  
 self.eye\_bitmap = wx.StaticBitmap(self, wx.ID\_ANY, self.hidden\_bitmap)  
 self.eye\_bitmap.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
 self.pass\_sizer.Add(self.eye\_bitmap)  
  
 entries.Add(name\_sizer, 0, wx.CENTER, 5)  
 entries.Add(self.pass\_sizer, 0, wx.CENTER, 10)  
  
 forgot\_password\_sizer.AddSpacer(28)  
 self.forgot\_password = TransparentText(self, -1, **"Forgot password"**, style=wx.TRANSPARENT\_WINDOW)  
 self.forgot\_password.SetFont(forgot\_password\_font)  
 self.forgot\_password.SetForegroundColour(**"#6787a4"**)  
 self.forgot\_password.Bind(wx.EVT\_LEFT\_DOWN, self.\_get\_user\_forgot\_password)  
 forgot\_password\_sizer.Add(self.forgot\_password)  
 entries.Add(forgot\_password\_sizer)  
  
 button\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 login\_button = wx.Button(self, label=**"LOG IN"**, size=(120, 40))  
 login\_button.SetFont(button\_font)  
 login\_button.Bind(wx.EVT\_BUTTON, self.on\_ok)  
  
 register\_button = wx.Button(self, label=**"REGISTER"**, size=(120, 40))  
 register\_button.SetFont(button\_font)  
 register\_button.Bind(wx.EVT\_BUTTON, self.register\_control)  
  
 button\_sizer.Add(login\_button, 0, wx.CENTER, 10)  
 button\_sizer.AddSpacer(80)  
 button\_sizer.Add(register\_button, 0, wx.CENTER, 10)  
  
 self.sizer.AddMany([(title, 0, wx.CENTER, 10),  
 (entries, 0, wx.CENTER, 5)])  
  
 self.sizer.AddSpacer(20)  
 self.sizer.Add(button\_sizer, 0, wx.CENTER, 5)  
  
 pub.subscribe(self.login\_ok, **"showLoginDialog"**)  
 pub.subscribe(self.verify\_ok, **"loginOk"**)  
 pub.subscribe(self.\_get\_details, **"detailsOk"**)  
 pub.subscribe(self.\_email\_ok, **"forgotPassEmailOk"**)  
 pub.subscribe(self.\_send\_forgot\_password\_request, **"showForgotPassDialog"**)  
  
 self.sizer.AddStretchSpacer()  
 self.sizer.Add(info\_button, 0, wx.ALL, 10)  
 info\_button.Bind(wx.EVT\_LEFT\_DOWN, self.show\_info)  
  
 self.SetSizer(self.sizer)  
 self.Layout()  
  
 self.Bind(wx.EVT\_PAINT, self.PaintBackgroundImage)  
  
 self.Hide()  
  
 @staticmethod  
 **def** show\_info(event):  
 *"""* **:param** *event: on click event* **:return***: shows about info dialog  
 """* info = wx.adv.AboutDialogInfo()  
 info.Name = **"GolDrive"** info.Version = **"1.4"** info.Description = **"""GolDrive lets you save your files securely in the cloud  
 so you can access them from anywhere!"""** info.Developers = [**"Reef Gold"**]  
 info.License = **"CC BY-ND"** info.SetCopyright(**"© 2024 Reef Gold. All rights reserved."**)  
  
 wx.adv.AboutBox(info)  
  
 **def** PaintBackgroundImage(self, evt):  
 *"""* **:param** *evt: paint event* **:return***: shows the background image  
 """* dc = wx.PaintDC(self)  
  
 bmp = wx.Bitmap(**rf"{**Settings.USER\_FILES\_PATH**}/bg.png"**)  
 dc.DrawBitmap(bmp, 0, 0)  
  
 **def** change\_visibility(self, event):  
 *"""* **:param** *event: on click event* **:return***: changes if you can see the password entered or not  
 """* open\_or\_closed = self.hidden\_bitmap **if not** self.is\_showing\_password **else** self.shown\_bitmap  
  
 position = self.passField.GetPosition()  
 size = self.passField.GetSize()  
  
 *# Create a new text control with the updated style* new\_style = wx.TE\_PASSWORD **if not** self.is\_showing\_password **else** wx.TE\_PROCESS\_ENTER  
 new\_textctrl = wx.TextCtrl(self, pos=position, size=size, style=new\_style | wx.TE\_PROCESS\_ENTER)  
 new\_textctrl.SetFont(self.entry\_font)  
  
 *# Copy the text from the existing text control to the new one* new\_textctrl.SetValue(self.passField.GetValue())  
  
 self.eye\_bitmap.SetBitmap(open\_or\_closed)  
 self.pass\_sizer.Replace(self.passField, new\_textctrl)  
 self.passField.Destroy()  
 self.passField = new\_textctrl  
 self.passField.Bind(wx.EVT\_TEXT\_ENTER, self.on\_ok)  
  
 self.is\_showing\_password = **not** self.is\_showing\_password  
  
 **def** register\_control(self, event):  
 *"""* **:param** *event: on click event* **:return***: shows the registration panel  
 """* self.parent.change\_screen(self, self.parent.register)  
  
 **def** login\_ok(self, email):  
 *"""* **:param** *email: email to send the verify email to* **:return***: sends the confirm mail request  
 """* con = ConfirmMailDialog(self, **"Verify"**)  
 result = con.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 values = con.GetValues()  
  
 msg = clientProtocol.pack\_verify\_login\_email\_request(email, \*values, self.parent.username)  
 self.comm.send(msg)  
  
 **def** verify\_ok(self):  
 *"""* **:return***: shows the users files  
 """* self.parent.username = self.username\_input  
 self.parent.files.title.SetLabel(self.parent.username.upper())  
  
 msg = clientProtocol.pack\_get\_details\_request(self.parent.username)  
 self.parent.files\_comm.send(msg)  
  
 self.parent.change\_screen(self, self.parent.files)  
  
 **def** on\_ok(self, event):  
 *"""* **:param** *event: on click event* **:return***: gets the input from the user and send login request  
 """* **with** open(**f"{**Settings.USER\_FILES\_PATH**}/Settings.png"**, **'rb'**) **as** f:  
 data = f.read()  
  
 self.parent.profilePhoto = data  
 self.parent.files.change\_settings\_to\_profile()  
  
 self.username\_input = self.nameField.GetValue()  
 password\_input = self.passField.GetValue()  
  
 self.nameField.SetValue(**""**)  
 self.passField.SetValue(**""**)  
  
 **if not** 0 < len(self.username\_input) <= 10:  
 self.parent.show\_pop\_up(**"Please enter a valid username and password."**, **"Error"**)  
 **else**:  
 self.parent.username = self.username\_input  
 msg2send = clientProtocol.pack\_login\_request(self.username\_input, password\_input)  
 self.comm.send(msg2send)  
  
 **def** \_get\_details(self, email, photo):  
 *"""* **:param** *email: email of user* **:param** *photo: profile photo* **:return***: changed the details of the user and changes the profile photo  
 """* self.parent.email = email  
 self.parent.profilePhoto = photo  
 wx.CallAfter(pub.sendMessage, **"changeSettingsToPhoto"**)  
  
 **def** \_get\_user\_forgot\_password(self, event):  
 *"""* **:param** *event: on click event* **:return***: send email verify request  
 """* dlg = wx.TextEntryDialog(self, **f'Enter username of user:'**, **'Verify'**, **''**)  
 result = dlg.ShowModal()  
 dlg.Destroy()  
  
 self.entered\_name = dlg.GetValue()  
  
 **if** result == wx.ID\_OK:  
 username = dlg.GetValue()  
  
 msg = clientProtocol.pack\_send\_email\_request(username)  
 self.comm.send(msg)  
  
 **def** \_email\_ok(self, email):  
 *"""* **:param** *email: the email adress that the email was sent to* **:return***: sends check code request  
 """* code\_dlg = wx.TextEntryDialog(self, **"Enter the 6-digit code sent to your mail:"**, **"Verify"**, **""**)  
 result = code\_dlg.ShowModal()  
 code\_dlg.Destroy()  
  
 **if** result == wx.ID\_OK:  
 code = code\_dlg.GetValue()  
  
 msg = clientProtocol.pack\_check\_code\_request(email, code)  
 self.comm.send(msg)  
  
 **def** \_send\_forgot\_password\_request(self):  
 *"""* **:return***: sends change password request  
 """* dlg = ForgotPasswordDialog(self, **"Enter new password"**)  
 result = dlg.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 values = dlg.GetValues()  
  
 msg = clientProtocol.pack\_forgot\_password\_request(self.entered\_name, \*values)  
 self.comm.send(msg)  
  
 dlg.Destroy()

mainPanel:  
**import** wx  
**from** pubsub **import** pub  
**from** .loginPanel **import** LoginPanel  
**from** .registerPanel **import** RegistrationPanel  
**from** .filesPanel **import** FilesPanel  
**from** settings **import** CurrentSettings **as** Settings  
**from** ClientFiles **import** clientProtocol  
  
  
**class** MainFrame(wx.Frame):  
 **def** \_\_init\_\_(self, comm, parent=**None**):  
 *"""* **:param** *comm: client comm object* **:param** *parent: panel parent  
 """* super(MainFrame, self).\_\_init\_\_(parent, title=**"GolDrive"**)  
 self.comm = comm  
 self.Maximize()  
  
 self.main\_panel = MainPanel(self, self.comm)  
  
 self.SetIcon(wx.Icon(**f"{**Settings.USER\_FILES\_PATH**}/logo.ico"**))  
  
 box = wx.BoxSizer(wx.VERTICAL)  
 box.Add(self.main\_panel, 1, wx.EXPAND)  
 *# arrange the frame* self.SetSizer(box)  
 self.Layout()  
 self.Show()  
  
  
**class** MainPanel(wx.Panel):  
 **def** \_\_init\_\_(self, parent, comm):  
 *"""* **:param** *parent:* **:param** *comm:  
 """* wx.Panel.\_\_init\_\_(self, parent)  
 self.frame = parent  
 self.comm = comm  
 self.files\_comm = **None** self.username = **None** self.email = **None** self.profilePhoto = **None** v\_box = wx.BoxSizer()  
 *# create object for each panel* self.login = LoginPanel(self, self.frame, self.comm)  
 self.register = RegistrationPanel(self, self.frame, self.comm)  
 self.files = FilesPanel(self, self.frame, self.comm)  
  
 v\_box.Add(self.login)  
 v\_box.Add(self.register)  
 v\_box.Add(self.files)  
  
 *# The first panel to show* self.login.Show()  
 self.SetSizer(v\_box)  
 self.Layout()  
  
 pub.subscribe(self.show\_pop\_up, **"showPopUp"**)  
 pub.subscribe(self.\_files\_comm\_update, **"updateFileComm"**)  
  
 **def** change\_screen(self, cur\_screen, screen):  
 *"""* **:param** *cur\_screen: screen showing panel object* **:param** *screen: screen to show panel object* **:return***: changes screen  
 """* cur\_screen.Hide()  
 screen.Show()  
  
 **if** screen == self.login:  
 msg = clientProtocol.disconnect\_alert()  
 self.comm.send(msg)  
  
 self.Layout()  
  
 **def** show\_pop\_up(self, text, title):  
 *"""* **:param** *text: text to show* **:param** *title: title of pop up* **:return***: shows pop up message dialog  
 """* dlg = wx.MessageDialog(self, text, title, wx.OK | wx.ICON\_INFORMATION)  
 dlg.ShowModal()  
 dlg.Destroy()  
  
 **def** \_files\_comm\_update(self, file\_comm):  
 *"""* **:param** *file\_comm: client file comm object* **:return***:  
 """* self.files\_comm = file\_comm

registerPanel:  
**import** wx  
**from** pubsub **import** pub  
**from** ClientFiles **import** clientProtocol  
**from** .customMenusAndDialogs **import** ConfirmMailDialog, TransparentText  
**from** settings **import** CurrentSettings **as** Settings  
  
  
**class** RegistrationPanel(wx.Panel):  
 **def** \_\_init\_\_(self, parent, frame, comm):  
 *"""* **:param** *parent: panel parent* **:param** *frame: frame parent* **:param** *comm: client comm object  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(1920, 1080), style=wx.SIMPLE\_BORDER)  
 self.frame = frame  
 self.comm = comm  
 self.parent = parent  
  
 self.username = **None** self.password = **None** self.email = **None** title\_font = wx.Font(68, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**, **"High Tower Text"**)  
 text\_font = wx.Font(30, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**)  
 self.entry\_font = wx.Font(20, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, **False**)  
 button\_font = wx.Font(15, wx.DEFAULT, wx.NORMAL, wx.BOLD)  
  
 hidden\_path = **f"{**Settings.USER\_FILES\_PATH**}/hidden-eye.png"** open\_path = **f"{**Settings.USER\_FILES\_PATH**}/open-eye.png"** self.is\_showing\_password = **False** self.hidden\_bitmap = wx.Bitmap(hidden\_path, wx.BITMAP\_TYPE\_ANY)  
 self.shown\_bitmap = wx.Bitmap(open\_path, wx.BITMAP\_TYPE\_ANY)  
  
 self.sizer = wx.BoxSizer(wx.VERTICAL)  
 entries = wx.BoxSizer(wx.VERTICAL)  
  
 self.sizer.AddSpacer(235)  
 title = TransparentText(self, -1, label=**"REGISTER"**, style=wx.TRANSPARENT\_WINDOW)  
 title.SetForegroundColour(wx.BLACK)  
 title.SetFont(title\_font)  
  
 name\_sizer = wx.BoxSizer(wx.VERTICAL)  
 name\_text = TransparentText(self, 1, label=**"Username: "**, style=wx.TRANSPARENT\_WINDOW)  
 name\_text.SetFont(text\_font)  
 self.nameField = wx.TextCtrl(self, -1, name=**"username"**, size=(650, 40))  
 self.nameField.SetFont(self.entry\_font)  
  
 name\_sizer.Add(name\_text, 0, wx.Center, 5)  
 name\_sizer.Add(self.nameField)  
 name\_sizer.AddSpacer(15)  
  
 self.pass\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 pass\_text = TransparentText(self, 1, label=**"Password: "**, style=wx.TRANSPARENT\_WINDOW)  
 pass\_text.SetFont(text\_font)  
 self.passField = wx.TextCtrl(self, -1, name=**"password"**, size=(650, 40),  
 style=wx.TE\_PASSWORD | wx.TE\_PROCESS\_ENTER)  
 self.passField.SetFont(self.entry\_font)  
  
 name\_sizer.Add(pass\_text)  
 self.pass\_sizer.AddSpacer(27)  
 self.pass\_sizer.Add(self.passField)  
 self.eye\_bitmap = wx.StaticBitmap(self, wx.ID\_ANY, self.hidden\_bitmap)  
 self.eye\_bitmap.Bind(wx.EVT\_LEFT\_DOWN, self.change\_visibility)  
 self.pass\_sizer.Add(self.eye\_bitmap)  
  
 email\_sizer = wx.BoxSizer(wx.VERTICAL)  
 email\_text = TransparentText(self, 1, label=**"Email: "**, style=wx.TRANSPARENT\_WINDOW)  
 email\_text.SetFont(text\_font)  
 self.emailField = wx.TextCtrl(self, -1, name=**"email"**, size=(650, 40))  
 self.emailField.SetFont(self.entry\_font)  
  
 email\_sizer.Add(email\_text, 0, wx.Center, 5)  
 email\_sizer.Add(self.emailField)  
 email\_sizer.AddSpacer(15)  
  
 entries.Add(name\_sizer, 0, wx.CENTER, 5)  
 entries.Add(self.pass\_sizer, 0, wx.CENTER, 5)  
 entries.Add(email\_sizer, 0, wx.CENTER, 10)  
  
 button\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 login\_button = wx.Button(self, label=**"BACK TO LOGIN"**)  
 login\_button.Bind(wx.EVT\_BUTTON, self.login\_control)  
 login\_button.SetFont(button\_font)  
  
 register\_button = wx.Button(self, label=**"REGISTER"**)  
 register\_button.Bind(wx.EVT\_BUTTON, self.on\_register)  
 register\_button.SetFont(button\_font)  
  
 *# self.SetTabOrder([self.nameField, self.passField, login\_button, register\_button])* button\_sizer.Add(register\_button, 0, wx.CENTER, 10)  
 button\_sizer.AddSpacer(80)  
 button\_sizer.Add(login\_button, 0, wx.CENTER, 10)  
  
 self.sizer.AddMany([(title, 0, wx.CENTER, 10),  
 (entries, 0, wx.CENTER, 5)])  
  
 self.sizer.AddSpacer(20)  
 self.sizer.Add(button\_sizer, 0, wx.CENTER, 5)  
  
 pub.subscribe(self.register\_ok, **"showRegisterDialog"**)  
 pub.subscribe(self.verify\_ok, **"registerOk"**)  
  
 self.Bind(wx.EVT\_PAINT, self.PaintBackgroundImage)  
  
 self.SetSizer(self.sizer)  
 self.Layout()  
  
 self.Hide()  
  
 **def** PaintBackgroundImage(self, evt):  
 *"""* **:param** *evt: on paint event* **:return***: paints the background image  
 """* dc = wx.PaintDC(self)  
  
 bmp = wx.Bitmap(**rf"{**Settings.USER\_FILES\_PATH**}\bg.png"**)  
 dc.DrawBitmap(bmp, 0, 0)  
  
 **def** change\_visibility(self, event):  
 *"""* **:param** *event: on click event* **:return***: changes if you can see the password entered or not  
 """* open\_or\_closed = self.hidden\_bitmap **if not** self.is\_showing\_password **else** self.shown\_bitmap  
  
 position = self.passField.GetPosition()  
 size = self.passField.GetSize()  
  
 *# Create a new text control with the updated style* new\_style = wx.TE\_PASSWORD **if not** self.is\_showing\_password **else** wx.TE\_PROCESS\_ENTER  
 new\_textctrl = wx.TextCtrl(self, pos=position, size=size, style=new\_style)  
 new\_textctrl.SetFont(self.entry\_font)  
  
 *# Copy the text from the existing text control to the new one* new\_textctrl.SetValue(self.passField.GetValue())  
  
 self.eye\_bitmap.SetBitmap(open\_or\_closed)  
 self.pass\_sizer.Replace(self.passField, new\_textctrl)  
 self.passField.Destroy()  
 self.passField = new\_textctrl  
  
 self.is\_showing\_password = **not** self.is\_showing\_password  
  
 **def** login\_control(self, event):  
 *"""* **:param** *event: on click event* **:return***: shows the login page panel  
 """* self.parent.change\_screen(self, self.parent.login)  
  
 **def** on\_register(self, event):  
 *"""* **:param** *event: on click event* **:return***: sends register request  
 """* username\_input = self.nameField.GetValue()  
 password\_input = self.passField.GetValue()  
 email\_input = self.emailField.GetValue()  
  
 self.nameField.SetValue(**""**)  
 self.passField.SetValue(**""**)  
 self.emailField.SetValue(**""**)  
  
 msg2send = clientProtocol.pack\_register\_request(username\_input, password\_input, email\_input)  
 self.comm.send(msg2send)  
  
 **def** register\_ok(self, username, password, email):  
 *"""* **:param** *username: username entered* **:param** *password: password entered* **:param** *email: email entered* **:return***: sends the code check request  
 """* self.username = username  
 self.password = password  
 self.email = email  
  
 con = ConfirmMailDialog(self, **"Verify"**)  
 result = con.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 values = con.GetValues()  
  
 msg = clientProtocol.pack\_verify\_register\_email\_request(username, password, email, \*values)  
 self.comm.send(msg)  
  
 **def** verify\_ok(self):  
 *"""* **:return***: shows the login page panel  
 """* self.parent.show\_pop\_up(**"User created successfully."**, **"Success"**)  
 self.parent.change\_screen(self, self.parent.login)

userPanel:  
**import** wx  
**from** pubsub **import** pub  
**import** io  
**from** ClientFiles **import** clientProtocol  
**from** .customMenusAndDialogs **import** ChangePasswordDialog, ProfileSettingsMenu, TransparentText  
**from** settings **import** CurrentSettings **as** Settings  
  
  
**class** UserPanel(wx.Panel):  
 **def** \_\_init\_\_(self, parent, frame, comm, files\_comm):  
 *"""* **:param** *parent: panel parent* **:param** *frame: frame parent* **:param** *comm: client comm object* **:param** *files\_comm: client file comm object  
 """* wx.Panel.\_\_init\_\_(self, parent, pos=wx.DefaultPosition, size=(1920, 1080), style=wx.SIMPLE\_BORDER)  
 self.frame = frame  
 self.comm = comm  
 self.files\_comm = files\_comm  
 self.parent = parent  
  
 button\_font = wx.Font(15, wx.DEFAULT, wx.NORMAL, wx.BOLD)  
  
 self.selected\_path = **""** self.sizer = wx.BoxSizer(wx.VERTICAL)  
 self.sizer.AddSpacer(2)  
  
 self.title = TransparentText(self, -1, label=**"SETTINGS"**)  
 font = wx.Font(65, wx.DECORATIVE, wx.NORMAL, wx.NORMAL, 0, **"High tower text"**)  
 self.title.SetFont(font)  
 self.sizer.Add(self.title, 0, wx.CENTER)  
  
 image = wx.Image(io.BytesIO(self.parent.profilePhoto), wx.BITMAP\_TYPE\_ANY)  
 image.Rescale(108, 108)  
  
 *# Convert the wx.Image to a wx.Bitmap* bitmap = wx.Bitmap(image)  
 self.settings\_img = wx.BitmapButton(self, wx.ID\_ANY, bitmap)  
  
 self.titleSizer = wx.BoxSizer(wx.VERTICAL)  
 self.usernameTitleSizer = wx.BoxSizer(wx.HORIZONTAL)  
 self.emailTitleSizer = wx.BoxSizer(wx.HORIZONTAL)  
 self.imageSizer = wx.BoxSizer(wx.HORIZONTAL)  
 self.userAndImageSizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 self.usernameTitle = TransparentText(self, -1, label=**f"Username: {**self.parent.username**}"**)  
 self.emailTitle = TransparentText(self, -1, label=**f"Email: {**self.parent.email**}"**)  
 title\_font = wx.Font(45, wx.DECORATIVE, wx.NORMAL, wx.NORMAL)  
 self.usernameTitle.SetFont(title\_font)  
 self.emailTitle.SetFont(title\_font)  
  
 *# adding padding to the text, so it doesn't start from the edge* self.usernameTitleSizer.AddSpacer(120)  
 self.usernameTitleSizer.Add(self.usernameTitle)  
  
 *# adding padding to the image, so it doesn't start from the edge* self.imageSizer.Add(self.settings\_img)  
 self.imageSizer.AddSpacer(17) *# Adjust the spacing if needed  
  
 # adding padding to the text, so it doesn't start from the edge* self.emailTitleSizer.AddSpacer(120)  
 self.emailTitleSizer.Add(self.emailTitle)  
  
 self.sizer.Add(self.imageSizer, 0, wx.ALIGN\_RIGHT)  
 self.sizer.Add(self.usernameTitleSizer)  
 self.sizer.AddSpacer(350)  
 self.sizer.Add(self.emailTitleSizer)  
  
 self.loginButton = wx.Button(self, label=**"BACK TO LOGIN"**)  
 self.loginButton.SetFont(button\_font)  
 self.filesButton = wx.Button(self, label=**"BACK TO FILES"**)  
 self.filesButton.SetFont(button\_font)  
 self.changeEmailButton = wx.Button(self, label=**"CHANGE EMAIL"**)  
 self.changeEmailButton.SetFont(button\_font)  
 self.changePasswordButton = wx.Button(self, label=**"CHANGE PASSWORD"**)  
 self.changePasswordButton.SetFont(button\_font)  
 self.changePhotoButton = wx.Button(self, label=**"CHANGE PROFILE PHOTO"**)  
 self.changePhotoButton.SetFont(button\_font)  
 self.deletePhotoButton = wx.Button(self, label=**"DELETE PROFILE PHOTO"**)  
 self.deletePhotoButton.SetFont(button\_font)  
  
 self.loginButton.Bind(wx.EVT\_BUTTON, self.login\_control)  
 self.filesButton.Bind(wx.EVT\_BUTTON, self.files\_control)  
 self.changeEmailButton.Bind(wx.EVT\_BUTTON, self.change\_email\_request)  
 self.changePasswordButton.Bind(wx.EVT\_BUTTON, self.change\_password\_request)  
 self.changePhotoButton.Bind(wx.EVT\_BUTTON, self.change\_photo\_request)  
 self.deletePhotoButton.Bind(wx.EVT\_BUTTON, self.delete\_photo\_request)  
 self.settings\_img.Bind(wx.EVT\_RIGHT\_DOWN, self.show\_settings\_menu)  
  
 self.buttons\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 self.buttons\_sizer.Add(self.filesButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.loginButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.changeEmailButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.changePasswordButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.changePhotoButton)  
 self.buttons\_sizer.AddSpacer(20)  
 self.buttons\_sizer.Add(self.deletePhotoButton)  
  
 self.sizer.AddSpacer(211)  
 self.sizer.Add(self.buttons\_sizer, 0, wx.CENTER)  
  
 self.SetSizer(self.sizer)  
 self.Layout()  
  
 pub.subscribe(self.\_change\_email, **"changeEmailOk"**)  
 pub.subscribe(self.\_change\_photo, **"changePhotoOk"**)  
  
 self.Bind(wx.EVT\_PAINT, self.PaintBackgroundImage)  
  
 self.Hide()  
  
 **def** PaintBackgroundImage(self, evt):  
 *"""* **:param** *evt: on paint event* **:return***: prints the background image  
 """* dc = wx.PaintDC(self)  
  
 bmp = wx.Bitmap(**f"{**Settings.USER\_FILES\_PATH**}/bg.png"**)  
 dc.DrawBitmap(bmp, 0, 0)  
  
 **def** login\_control(self, event):  
 *"""* **:param** *event: on click event* **:return***: shows the login page panel  
 """* self.parent.change\_screen(self, self.parent.login)  
  
 **def** files\_control(self, event):  
 *"""* **:param** *event: on click event* **:return***: shows the users files panel  
 """* self.parent.change\_screen(self, self.parent.files)  
  
 **def** change\_email\_request(self, event):  
 *"""* **:param** *event: on click event* **:return***: sends email change request  
 """* email\_dlg = wx.TextEntryDialog(self, **f'What new email to change to?'**, **'Confirmation'**, **''**)  
 result = email\_dlg.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 email = email\_dlg.GetValue()  
 email\_dlg.Destroy()  
  
 split\_email = email.split(**'@'**)  
 *# checking if email is valid* **if '@' in** email **and** len(split\_email) == 2 **and** split\_email[0] **and** split\_email[1]:  
 msg = clientProtocol.pack\_send\_verify\_request(email)  
 self.comm.send(msg)  
  
 verify\_dlg = wx.TextEntryDialog(self, **f'Which 6-digit code did you receive in your mail?'**, **'Verify'**, **''**)  
 result = verify\_dlg.ShowModal()  
  
 *# sends the request to check if the code entered is correct* **if** result == wx.ID\_OK:  
 msg = clientProtocol.pack\_change\_email\_request(self.parent.username, email, verify\_dlg.GetValue())  
 self.comm.send(msg)  
  
 verify\_dlg.Destroy()  
 **else**:  
 self.parent.show\_pop\_up(**"Please enter a valid email address."**, **"Error"**)  
  
 **else**:  
 email\_dlg.Destroy()  
  
 **def** \_change\_email(self, email):  
 *"""* **:param** *email: email to change to* **:return***: changes email of user  
 """* self.parent.email = email  
 self.emailTitle.SetLabel(**f"Email: {**email**}"**)  
 self.Layout()  
  
 self.parent.show\_pop\_up(**f"Changed email to {**email**} successfully."**, **"Success"**)  
  
 **def** change\_password\_request(self, event):  
 *"""* **:param** *event: on click event* **:return***: sends change password request  
 """* password\_dlg = ChangePasswordDialog(self, **"Confirmation"**)  
 result = password\_dlg.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 values = password\_dlg.GetValues()  
  
 msg = clientProtocol.pack\_change\_password\_request(self.parent.username, \*values)  
 self.comm.send(msg)  
 password\_dlg.Destroy()  
  
 **def** change\_photo\_request(self, event):  
 *"""* **:param** *event: on click event* **:return***: sends change photo request  
 """* image\_types = [**"apng"**, **"avif"**, **"gif"**, **"jpg"**, **"jpeg"**, **"jfif"**, **"pjpeg"**, **"pjp"**, **"png"**, **"svg"**, **"webp"**, **"bmp"**, **"ico"**,  
 **"cur"**, **"tif"**, **"tiff"**]  
  
 what\_to\_show = ([**f'\*.{**image\_type**}' for** image\_type **in** image\_types])  
 wildcard = **f"Image Files ({';'**.join(what\_to\_show)**})|{';'**.join(what\_to\_show)**}"** dlg = wx.FileDialog(self, **"Choose a file"**, wildcard=wildcard, style=wx.DD\_DEFAULT\_STYLE)  
 result = dlg.ShowModal()  
  
 **if** result == wx.ID\_OK:  
 self.selected\_path = dlg.GetPath().replace(**'\\'**, **'/'**)  
 self.files\_comm.send\_file(4, self.selected\_path, self.parent.username)  
  
 **def** \_change\_photo(self, data):  
 *"""* **:param** *data: data of the photo* **:return***: changes the users photo  
 """* self.parent.profilePhoto = data  
  
 image = wx.Image(io.BytesIO(self.parent.profilePhoto), wx.BITMAP\_TYPE\_ANY)  
 image.Rescale(108, 108)  
  
 *# Convert the wx.Image to a wx.Bitmap* bitmap = wx.Bitmap(image)  
  
 self.parent.files.settings\_img.SetBitmap(bitmap)  
 self.settings\_img.SetBitmap(bitmap)  
  
 **def** delete\_photo\_request(self, event):  
 *"""* **:param** *event: on click event* **:return***: sends delete profile photo request  
 """* verify\_dlg = wx.MessageDialog(self, **f'Are you sure you want to delete your profile photo?'**,  
 **'Confirmation'**, wx.YES\_NO | wx.ICON\_QUESTION)  
 result = verify\_dlg.ShowModal()  
  
 **if** result == wx.ID\_YES:  
 msg = clientProtocol.pack\_delete\_profile\_photo\_request(self.parent.username)  
 self.files\_comm.send(msg)  
  
 **def** show\_settings\_menu(self, event):  
 *"""* **:param** *event: on click event* **:return***: shows the settings menu  
 """* self.PopupMenu(ProfileSettingsMenu(self))

DB:  
**import** sqlite3  
**import** encryption  
  
  
**class** DB:  
 **def** \_\_init\_\_(self):  
 *"""  
 creates the table  
 """* self.DB\_name = **"GolDrive.sql"** self.conn = **None** self.curr = **None** self.\_create\_data\_base()  
  
 **def** \_create\_data\_base(self):  
 *"""  
 create table* **:return***:  
 """  
  
 # enabling changes from different threads* self.conn = sqlite3.connect(self.DB\_name, check\_same\_thread=**False**)  
 self.curr = self.conn.cursor()  
 create\_users = **"""  
 CREATE TABLE IF NOT EXISTS users(  
 username VARCHAR(10) PRIMARY KEY,   
 password VARCHAR(64),   
 email VARCHAR(50));  
 """** create\_users\_ips = **"""  
 CREATE TABLE IF NOT EXISTS ipByUser(  
 username VARCHAR(10),  
 ip VARCHAR(15));  
 """** self.curr.execute(create\_users)  
 self.curr.execute(create\_users\_ips)  
 self.conn.commit()  
  
 **def** username\_exist(self, username):  
 *"""* **:param** *username: username* **:return***: True or False if username is in table  
 """* sql = **"SELECT username FROM users WHERE username = ?"** self.curr.execute(sql, (username,))  
 **return** self.curr.fetchone()  
  
 **def** add\_user(self, username, password, email):  
 *"""* **:param** *username: username* **:param** *password: password* **:param** *email: email* **:return***: adds user to table  
 """* flag = 1  
 **if not** self.username\_exist(username):  
 sql = **"INSERT INTO users VALUES (?,?,?)"** self.curr.execute(sql, (username, encryption.hash\_msg(password), email))  
 self.conn.commit()  
  
 flag = 0  
 **return** flag  
  
 **def** delete\_user(self, username):  
 *"""* **:param** *username: username* **:return***: delets user  
 """* flag = 1  
 **if** self.username\_exist(username):  
 sql = **"DELETE FROM users WHERE username = ?"** self.curr.execute(sql, (username,))  
 self.conn.commit()  
  
 flag = 0  
 **return** flag  
  
 **def** get\_password(self, username):  
 *"""* **:param** *username: username* **:return***: users password  
 """* **if** self.username\_exist(username):  
 sql = **"SELECT password FROM users WHERE username = ?"** self.curr.execute(sql, (username,))  
 password, = self.curr.fetchone()  
 **return** password  
  
 **def** get\_email(self, username):  
 *"""* **:param** *username: username* **:return***: users email  
 """* **if** self.username\_exist(username):  
 sql = **"SELECT email FROM users WHERE username = ?"** self.curr.execute(sql, (username,))  
 email, = self.curr.fetchone()  
 **return** email  
  
 **def** change\_username(self, username, newname):  
 *"""* **:param** *username: username* **:param** *newname: name to change to* **:return***: changes username  
 """* flag = 1  
 **if** self.username\_exist(username):  
 sql = **f"UPDATE users SET username = ? WHERE username = ?"** self.curr.execute(sql, (newname, username,))  
 self.conn.commit()  
  
 flag = 0  
 **return** flag  
  
 **def** change\_password(self, username, password):  
 *"""* **:param** *username: username* **:param** *password: password to change to* **:return***: changes password  
 """* flag = 1  
 **if** self.username\_exist(username):  
 sql = **f"UPDATE users SET password = ? WHERE username = ?"** self.curr.execute(sql, (password, username,))  
 self.conn.commit()  
  
 flag = 0  
 **return** flag  
  
 **def** change\_email(self, username, email):  
 *"""* **:param** *username: username* **:param** *email: email to change to* **:return***: changes email  
 """* flag = 1  
 **if** self.username\_exist(username):  
 sql = **f"UPDATE users SET email = ? WHERE username = ?"** self.curr.execute(sql, (email, username,))  
 self.conn.commit()  
  
 flag = 0  
 **return** flag  
  
 **def** get\_ips\_of\_user(self, username):  
 *"""* **:param** *username: username of user* **:return***: all ips the user asked to not be asked again on  
 """* sql = **"SELECT ip FROM ipByUser WHERE username = ?"** self.curr.execute(sql, (username,))  
  
 ips = self.curr.fetchall()  
 remembered\_ips = []  
 **for** ipTuple **in** ips:  
 **for** ip **in** ipTuple:  
 remembered\_ips.append(ip)  
  
 **return** remembered\_ips  
  
 **def** add\_remembered\_ip(self, username, ip):  
 *"""* **:param** *username: username of user* **:param** *ip: ip of computer* **:return***: adds the username with the ip to be remembered  
 """* sql = **"INSERT INTO ipByUser VALUES (?,?)"** self.curr.execute(sql, (username, ip))  
 self.conn.commit()  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 s = DB()  
 s.add\_remembered\_ip(**"reef"**, **"10.0.0.12"**)

portsHandler:  
**class** PortsHandler:  
 unused\_ports = (port **for** port **in** range(1235, 6235))  
  
 @staticmethod  
 **def** get\_next\_port():  
 *"""* **:return***: returns the next port in the generator  
 """* **return** next(PortsHandler.unused\_ports)

serverComm:  
**import** os.path  
**import** socket  
**import** select  
**import** threading  
**import** queue  
**from** settings **import** CurrentSettings **as** Settings  
**import** serverProtocol  
**import** encryption  
  
  
**class** ServerComm:  
 **def** \_\_init\_\_(self, port, recv\_q, msg\_len\_bytes):  
 *"""* **:param** *port* **:param** *recv\_q  
 """* self.port = port  
 self.recv\_q = recv\_q  
 self.serverSocket = socket.socket()  
 self.openClients = {} *# [socket]:[ip, key]* self.isRunning = **False** self.msgLenBytes = msg\_len\_bytes  
  
 threading.Thread(target=self.\_main\_loop).start()  
  
 **def** \_main\_loop(self):  
 *"""* **:return***: main loop  
 """* self.serverSocket.bind((**'0.0.0.0'**, self.port))  
 self.serverSocket.listen(3)  
 self.isRunning = **True  
 while** self.isRunning:  
 rlist, wlist, xlist = select.select([self.serverSocket] + list(self.openClients.keys()),  
 list(self.openClients.keys()), [], 0.03)  
  
 **for** currSocket **in** rlist:  
 **if** currSocket **is** self.serverSocket:  
 client, addr = self.serverSocket.accept()  
 print(**f"{**addr[0]**} - Connected"**)  
  
 threading.Thread(target=self.\_change\_key, args=(client, addr[0],)).start()  
 **continue  
  
 try**:  
 data\_len = int(currSocket.recv(self.msgLenBytes).decode())  
 data = currSocket.recv(data\_len)  
 enc\_obj = self.openClients[currSocket][1]  
  
 decrypted\_data = enc\_obj.dec\_msg(data)  
 **except** Exception **as** e:  
 print(**"main server in server comm "**, str(e))  
 self.\_handle\_disconnect(currSocket)  
  
 **else**:  
  
 **if** decrypted\_data == **""**:  
 self.\_handle\_disconnect(currSocket)  
  
 **else**:  
 **if** self.port == Settings.SERVERPORT:  
 self.recv\_q.put((self.openClients[currSocket][0], decrypted\_data))  
  
 *# files server* **else**:  
 opcode, params = serverProtocol.unpack\_message(decrypted\_data)  
  
 **if** opcode == **"12" or** opcode == **"04"**:  
 self.recv\_file(opcode, currSocket, \*params)  
 **else**:  
 self.recv\_q.put((self.openClients[currSocket][0], decrypted\_data))  
  
 **def** \_change\_key(self, client, ip):  
 *"""* **:param** *client: client socket* **:param** *ip: client ip* **:return***: exchanges a symmetrical keys in order to get aes key  
 """* privateA, a = encryption.get\_dh\_factor()  
 **try**:  
 client.send((str(a).zfill(10)).encode())  
 b = int(client.recv(10).decode())  
 **except** Exception **as** e:  
 print(**"in change key "**, str(e))  
 client.close()  
 **else**:  
 crypto = encryption.create\_symmetry\_key(privateA, b)  
 self.openClients[client] = [ip, crypto]  
 print(**f"{**ip**} - complete change key"**)  
  
 **def** \_handle\_disconnect(self, sock):  
 **if** sock **in** self.openClients.keys():  
 print(**f"{**self.openClients[sock][0]**} - Disconnected"**)  
 **del** self.openClients[sock]  
 sock.close()  
  
 **def** \_find\_socket\_by\_ip(self, ip):  
 *"""* **:param** *ip:* **:return***: return socket with IP ip  
 """* sock = **None  
 for** soc **in** self.openClients.keys():  
 **if** self.openClients[soc][0] == ip:  
 sock = soc  
 **break  
 return** sock  
  
 **def** send(self, ip, data):  
 *"""* **:param** *ip: ip to send to* **:param** *data: data of message to send* **:return***: sends msg to the socket with right ip  
 """* **if** self.\_running\_status():  
 client = self.\_find\_socket\_by\_ip(ip)  
 **if** client:  
 enc\_obj = self.openClients[client][1]  
 encrypt\_data = enc\_obj.enc\_msg(data)  
 *# the length of the encrypted message* data\_len = str(len(encrypt\_data)).zfill(self.msgLenBytes).encode()  
 **try**:  
 client.send(data\_len + encrypt\_data)  
 **except** Exception **as** e:  
 print(**"server com - send"**, str(e))  
 self.\_handle\_disconnect(client)  
  
 **def** \_close\_server(self):  
 *"""* **:return***: changes server status to close it  
 """* self.isRunning = **False  
  
 def** \_running\_status(self):  
 *"""* **:return***: is server running  
 """* **return** self.isRunning  
  
 **def** recv\_file(self, opcode, client, file\_path, file\_len):  
 *"""* **:param** *opcode: opcode of the message* **:param** *client: the socket of the client* **:param** *file\_path: the file path in the server* **:param** *file\_len: the length of the file* **:return***: adds message to socket for logic to handle  
 """* data = bytearray()  
 file\_len = int(file\_len)  
 ip = self.openClients[client][0]  
  
 **try**:  
 **while** len(data) < file\_len:  
 slices = file\_len - len(data)  
 **if** slices > 1024:  
 data.extend(client.recv(1024))  
 **else**:  
 data.extend(client.recv(slices))  
 **break  
 except** Exception **as** e:  
 **if** opcode == **"12"**:  
 self.recv\_q.put((ip, (**"12"**, file\_path, **None**)))  
 **else**:  
 self.recv\_q.put((ip, (**"04"**, file\_path, **None**)))  
 print(**"main server in recv file comm "**, str(e))  
 self.\_handle\_disconnect(client)  
 **else**:  
 decData = self.openClients[client][1].dec\_msg(data)  
 **if** opcode == **"12"**:  
 self.recv\_q.put((ip, (**"12"**, file\_path, decData)))  
 **else**:  
 self.recv\_q.put((ip, (**"04"**, file\_path, decData)))  
  
 **def** send\_file(self, client\_ip, params):  
 *"""* **:param** *client\_ip: clients ip* **:param** *params: the params to use* **:return***:  
 """* client\_socket = self.\_find\_socket\_by\_ip(client\_ip)  
  
 **try**:  
 opcode = params[0]  
 **except** Exception **as** e:  
 print(**"in send file - "**, str(e))  
  
 **else**:  
 **if** client\_socket:  
 status = 0  
  
 *# applying correct variables according to the opcode* username = **"" if** opcode != **"04" and** opcode != **"21" else** params[1]  
 path = **"" if** opcode != **"11" and** opcode != **"20" else** params[1]  
 selected\_path = **"" if** opcode != **"11" else** params[2]  
 email = **"" if** opcode != **"21" else** params[2]  
  
 *# if the user doesn't have a profile photo we give him the default photo.  
 # its d since username cant be 1 char long* **if not** os.path.isfile(**f"{**Settings.USER\_PROFILE\_PHOTOS**}/{**username**}.png"**):  
 username = **'d'** path\_to\_read = **f"{**Settings.SERVER\_FILES\_PATH**}/{**path**}" if** opcode != **"21" and** opcode != **"04" else** \  
 **f"{**Settings.USER\_PROFILE\_PHOTOS**}/{**username**}.png"  
  
 try**:  
 **with** open(path\_to\_read, **'rb'**) **as** f:  
 data = f.read()  
 **except** Exception **as** e:  
 print(str(e))  
 *# applying error message* status = 1  
 **if** opcode == **"04"**:  
 msg = serverProtocol.pack\_change\_photo\_response(status, 0)  
 **elif** opcode == **"11"**:  
 msg = serverProtocol.pack\_file\_download\_response(status, 0, path, selected\_path)  
 **elif** opcode == **"21"**:  
 msg = serverProtocol.pack\_get\_details\_response(status, **None**, 0)  
 **else**:  
 msg = serverProtocol.pack\_open\_file\_response(status, path, 0)  
  
 self.send(client\_ip, msg)  
  
 **else**:  
 *# sending the correct message according to protocol* cryptFile = self.openClients[client\_socket][1].enc\_msg(data)  
 **if** opcode == **"04"**:  
 msg = serverProtocol.pack\_change\_photo\_response(status, len(cryptFile))  
 **elif** opcode == **"11"**:  
 msg = serverProtocol.pack\_file\_download\_response(status, len(cryptFile), path, selected\_path)  
 **elif** opcode == **"21"**:  
 msg = serverProtocol.pack\_get\_details\_response(status, len(cryptFile), email)  
 **else**:  
 msg = serverProtocol.pack\_open\_file\_response(status, path, len(cryptFile))  
  
 self.send(client\_ip, msg)  
 **try**:  
 client\_socket.send(cryptFile)  
 **except** Exception **as** e:  
 print(str(e))  
 self.\_handle\_disconnect(client\_socket)  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 msgQ = queue.Queue()  
 server = ServerComm(1000, msgQ, 2)

serverLogic:  
**import** os.path  
**import** queue  
**import** threading  
**import** portsHandler  
**import** serverComm  
**import** serverProtocol  
**from** settings **import** CurrentSettings **as** Settings  
**import** DB  
**import** encryption  
**import** sFileHandler  
**import** shutil  
**import** smtplib  
**import** secrets  
**from** email.mime.text **import** MIMEText  
**from** email.mime.multipart **import** MIMEMultipart  
  
  
**def** main\_loop():  
 *"""* **:return***: the main loop of the server  
 """* msg\_q = queue.Queue()  
  
 main\_server = serverComm.ServerComm(Settings.SERVERPORT, msg\_q, 4)  
 threading.Thread(target=\_handle\_messages, args=(main\_server, msg\_q,)).start()  
  
  
**def** \_handle\_messages(main\_server, msg\_q):  
 *"""* **:param** *main\_server: the server object* **:param** *msg\_q: the queue of messages to handle* **:return***: takes the message out the message queue and runs the according function  
 """* my\_db = DB.DB()  
 user\_by\_ip = {}  
 code\_dic = {}  
  
 recv\_commands = {**"01"**: \_handle\_registration, **"02"**: \_handle\_login, **"05"**: \_handle\_change\_email,  
 **"06"**: \_handle\_change\_password, **"07"**: \_send\_email, **"08"**: \_handle\_rename\_file,  
 **"09"**: \_handle\_share\_file, **"10"**: \_handle\_delete\_file, **"13"**: \_handle\_create\_dir,  
 **"17"**: \_handle\_send\_email, **"18"**: \_handle\_move\_file, **"19"**: \_handle\_paste\_file,  
 **"23"**: \_handle\_email\_login, **"24"**: \_handle\_email\_register, **"25"**: \_handle\_check\_email,  
 **"26"**: \_handle\_forgot\_password, **"27"**: \_handle\_zip\_folder, **"28"**: \_handle\_disconnect}  
  
 **while True**:  
 ip, data = msg\_q.get()  
 protocol\_num, params = serverProtocol.unpack\_message(data)  
 **if** protocol\_num == **"01" or** protocol\_num == **"05" or** protocol\_num == **"07" or** protocol\_num == **"02" or** \  
 protocol\_num == **"17" or** protocol\_num == **"23" or** protocol\_num == **"24" or** protocol\_num == **"25"**:  
 params.insert(0, code\_dic)  
  
 **if** protocol\_num == **"02" or** protocol\_num == **"09" or** protocol\_num == **"23" or** protocol\_num == **"28"**:  
 params.insert(0, user\_by\_ip)  
  
 **if** protocol\_num **in** recv\_commands.keys():  
 recv\_commands[protocol\_num](main\_server, my\_db, ip, \*params)  
 **else**:  
 print(**f"{**protocol\_num**} not in recv commands."**)  
  
  
**def** \_handle\_registration(main\_server, db, client\_ip, code\_dic, username, password, email):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *username: the username of user* **:param** *password: the password of user* **:param** *email: the email of user* **:return***: adds the user to the database and return 0 or 1 if it managed to add the user  
 """* status = 2  
  
 **if** 10 >= len(username) >= 4 **and** len(password) >= 4:  
 **if not** db.username\_exist(username):  
 status = 0  
 \_send\_email(main\_server, db, client\_ip, code\_dic, email)  
 **else**:  
 status = 1  
  
 msg = serverProtocol.pack\_register\_response(status, username, password, email)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_email\_register(main\_server, db, client\_ip, code\_dic, username, password, email, code, dont\_ask\_again):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary containing the code sent to an email* **:param** *username: the username of the client* **:param** *password: the password of the client* **:param** *email: the email of the client* **:param** *code: the code entered* **:param** *dont\_ask\_again: does the user want to verify his email in his computer* **:return***: try's adding the user to the database and sends response to the client  
 """* ans = 2  
  
 **if** email **in** code\_dic:  
 **if** code\_dic[email] == code:  
 **if** 4 <= len(username) <= 10 **and** len(password) >= 4:  
 ans = db.add\_user(username, password, email)  
  
 **if** ans == 0 **and** dont\_ask\_again == **"True"**:  
 db.add\_remembered\_ip(username, client\_ip)  
  
 msg = serverProtocol.pack\_register\_verify\_response(ans)  
 main\_server.send(client\_ip, msg)  
  
 **if** ans == 0:  
 **if not** os.path.isdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"**):  
 os.mkdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"**)  
 os.mkdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}/@#$SHAREDFILES$#@"**)  
  
  
**def** \_handle\_login(main\_server, db, client\_ip, users\_by\_ip, code\_dic, username, password):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *username: the username of user* **:param** *password: the password of user* **:return***: adds the user to the database and return 0 or 1 if the username and password matched an existing user  
 """* status = 1  
 email = db.get\_email(username)  
  
 **if** encryption.hash\_msg(password) == db.get\_password(username) **and not** any(username == user  
 **for** user **in** users\_by\_ip.values()):  
 status = 0  
  
 msg = serverProtocol.pack\_login\_response(status, email)  
  
 **if** status == 0:  
 remembered\_ips = db.get\_ips\_of\_user(username)  
  
 **if** client\_ip **in** remembered\_ips:  
 \_create\_files\_thread(main\_server, db, client\_ip, users\_by\_ip, username)  
 msg = serverProtocol.pack\_login\_verify\_response(status)  
 **else**:  
 \_send\_email(main\_server, db, client\_ip, code\_dic, email)  
  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_email\_login(main\_server, db, client\_ip, users\_by\_ip, code\_dic, email, code, dont\_ask\_again, username):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary containing the code sent to an email* **:param** *email: the email of the client* **:param** *code: the code entered* **:param** *dont\_ask\_again: does the user want to verify his email in his computer* **:param** *username: the username of the client* **:return***: checks if user entered the correct code sent to his email and sends response to client  
 """* status = 1  
  
 **if** email **in** code\_dic:  
 **if** code\_dic[email] == code:  
 status = 0  
  
 \_create\_files\_thread(main\_server, db, client\_ip, users\_by\_ip, username)  
  
 **if** dont\_ask\_again == **"True"**:  
 db.add\_remembered\_ip(username, client\_ip)  
  
 msg = serverProtocol.pack\_login\_verify\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_create\_files\_thread(main\_server, db, client\_ip, users\_by\_ip, username):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the client ip* **:param** *users\_by\_ip: dictionary containing users by their ips* **:param** *username: username of client* **:return***: gets the next available port which is used to upload and download files  
 """* users\_by\_ip[client\_ip] = username  
 port\_to\_give = portsHandler.PortsHandler.get\_next\_port()  
 files\_q = queue.Queue()  
 files\_server = serverComm.ServerComm(port\_to\_give, files\_q, 10)  
 threading.Thread(target=handle\_files, args=(files\_server, files\_q, db,)).start()  
 port\_msg = serverProtocol.pack\_upload\_port(port\_to\_give)  
 main\_server.send(client\_ip, port\_msg)  
  
 main\_server.send(client\_ip, serverProtocol.pack\_files\_message(username))  
  
  
**def** handle\_files(files\_server, files\_q, db):  
 *"""* **:param** *files\_server: the files server object* **:param** *files\_q: queue with messages waiting for handle* **:param** *db: the database* **:return***: starts the correct function according to the opcode  
 """* recv\_commands = {**"04"**: \_handle\_change\_photo, **"11"**: \_handle\_download\_file, **"12"**: \_handle\_upload\_file,  
 **"20"**: \_handle\_open\_file, **"21"**: \_handle\_get\_details, **"22"**: \_handle\_delete\_profile\_photo}  
  
 **while True**:  
 ip, data = files\_q.get()  
  
 **if** len(data) != 0:  
 **if** data[0] == **"12" or** data[0] == **"04"**:  
 protocol\_num, \*params = data  
 **else**:  
 protocol\_num, params = serverProtocol.unpack\_message(data)  
  
 **if** protocol\_num == **"21"**:  
 params.insert(0, db)  
  
 **if** protocol\_num **in** recv\_commands.keys():  
 recv\_commands[protocol\_num](files\_server, ip, \*params)  
  
  
**def** \_handle\_delete\_file(main\_server, db, client\_ip, path):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *path: the path of the file to delete* **:return***: deletes the file and return the message by the protocol  
 """* path\_on\_server = **f"{**Settings.SERVER\_FILES\_PATH**}/{**path**}"** status = 0  
  
 **if** os.path.isdir(path\_on\_server):  
 **try**:  
 shutil.rmtree(path\_on\_server)  
 **except** Exception **as** e:  
 print(**f"in delete file - {**str(e)**}"**)  
 status = 1  
  
 **elif** os.path.isfile(path\_on\_server):  
 **try**:  
 os.remove(path\_on\_server)  
 **except** Exception **as** e:  
 print(**f"in delete file - {**str(e)**}"**)  
 status = 1  
 **else**:  
 status = 1  
  
 msg = serverProtocol.pack\_delete\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_rename\_file(main\_server, db, client\_ip, path, new\_name):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *path: the path of the file to rename* **:param** *new\_name: the new name of the file* **:return***: renames the file and return the message by the protocol  
 """* status = sFileHandler.rename\_file(**f"{**Settings.SERVER\_FILES\_PATH**}/{**path**}"**, new\_name)  
 msg = serverProtocol.pack\_rename\_file\_response(status, new\_name)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_download\_file(main\_server, client\_ip, path, selected\_path):  
 *"""* **:param** *main\_server: the server object* **:param** *client\_ip: the clients ip* **:param** *path: the path of the file to download* **:param** *selected\_path: the path where the user wants to download the file into* **:return***: returns the data of the file  
 """* params = (**"11"**, path, selected\_path)  
 main\_server.send\_file(client\_ip, params)  
  
  
**def** \_handle\_open\_file(main\_server, client\_ip, path):  
 *"""* **:param** *main\_server: the server object* **:param** *client\_ip: the clients ip* **:param** *path: the path of the file to download* **:return***: returns the data of the file  
 """* params = (**"20"**, path)  
 main\_server.send\_file(client\_ip, params)  
  
  
**def** \_handle\_upload\_file(main\_server, client\_ip, path, data):  
 *"""* **:param** *main\_server: the server object* **:param** *client\_ip: the clients ip* **:param** *path: the path of the fle uploaded* **:param** *data: the data of the file* **:return***: uploads the file and sends response to the client  
 """* status = 0  
 **try**:  
 **with** open(**f"{**Settings.SERVER\_FILES\_PATH**}/{**path**}"**, **'wb' if** type(data) == bytes **else 'w'**) **as** f:  
 f.write(data)  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
  
 msg = serverProtocol.pack\_file\_upload\_response(status, path)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_create\_dir(main\_server, db, client\_ip, path):  
 *"""* **:param** *main\_server: server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *path: the path of the directory to create* **:return***: creates dir and sends response to the client  
 """* status = 0  
 **try**:  
 os.mkdir(**f"{**Settings.SERVER\_FILES\_PATH**}/{**path**}"**)  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
 msg = serverProtocol.pack\_create\_folder\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_share\_file(main\_server, db, client\_ip, users\_by\_ip, path, username):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *users\_by\_ip: dictionary of the user logged in according to the ip* **:param** *path: path of the file to share* **:param** *username: the username of the to share* **:return***: shares the file and sends response to the user  
 """* status = 0  
 user\_who\_shared = path.split(**'/'**)[0]  
 path\_to\_add = **f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}/@#$SHAREDFILES$#@/{**user\_who\_shared**}"  
  
 if** db.username\_exist(username):  
 **if not** os.path.isdir(path\_to\_add):  
 **try**:  
 os.mkdir(path\_to\_add)  
 **except** Exception **as** e:  
 print(str(e))  
  
 **try**:  
 shutil.copy(**f"{**Settings.SERVER\_FILES\_PATH**}/{**path**}"**,  
 **f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}/@#$SHAREDFILES$#@/"  
 f"{**user\_who\_shared**}"**)  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
  
 **else**:  
 status = 1  
  
 msg = serverProtocol.pack\_share\_response(status)  
 main\_server.send(client\_ip, msg)  
  
 ips\_to\_share = [key **for** key, value **in** users\_by\_ip.items() **if** value == username]  
  
 *# sends an update to all the computer logged on to the user shared* **for** ip **in** ips\_to\_share:  
 msg = serverProtocol.pack\_add\_shared\_file(path)  
 main\_server.send(ip, msg)  
  
  
**def** \_handle\_move\_file(main\_server, db, client\_ip, src, dst):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *src: the src path* **:param** *dst: the dst path* **:return***: moves the file from src to dst and sends response to the client  
 """* status = 0  
 **try**:  
 shutil.move(**f"{**Settings.SERVER\_FILES\_PATH**}/{**src**}"**, **f"{**Settings.SERVER\_FILES\_PATH**}/{**dst**}"**)  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
  
 msg = serverProtocol.pack\_move\_file\_response(status, dst)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_paste\_file(main\_server, db, client\_ip, src, dst):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *src: the src path* **:param** *dst: the dst path* **:return***: sends paste file request from src to dst according to the protocol  
 """* status = 0  
 full\_src\_path = **f"{**Settings.SERVER\_FILES\_PATH**}/{**src**}"** full\_dst\_path = **f"{**Settings.SERVER\_FILES\_PATH**}/{**dst**}"  
  
 try**:  
 shutil.copy(full\_src\_path, full\_dst\_path)  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
  
 msg = serverProtocol.pack\_paste\_file\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_get\_details(main\_server, client\_ip, db, username):  
 email = db.get\_email(username)  
  
 params = (**'21'**, username, email)  
 main\_server.send\_file(client\_ip, params)  
  
  
**def** \_send\_email(main\_server, my\_db, client\_ip, code\_dic, email):  
 *"""* **:param** *main\_server: the server object* **:param** *my\_db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary with the code sent and the email sent to* **:param** *email: the email the code was sent to* **:return***: sends verify code email to the email  
 """  
  
 # email to send from* sender\_email = **"goldriveauth@gmail.com"** receiver\_email = email  
 password = **"pukb pfll qplp gukh"** message = MIMEMultipart()  
 message[**"From"**] = sender\_email  
 message[**"To"**] = receiver\_email  
 message[**"Subject"**] = **"Verify email"** *# the verify code* verify\_code = secrets.SystemRandom().randint(100000, 999999)  
 code\_dic[email] = str(verify\_code)  
 body = **f"The code to verify is {**verify\_code**}."** message.attach(MIMEText(body, **"plain"**))  
  
 **if** receiver\_email:  
 *# connect to the SMTP server* **try**:  
 smtp\_server = smtplib.SMTP(**"smtp.gmail.com"**, 587)  
 smtp\_server.starttls()  
 smtp\_server.login(sender\_email, password)  
  
 *# send the email* smtp\_server.sendmail(sender\_email, receiver\_email, message.as\_string())  
  
 **except** smtplib.SMTPException **as** e:  
 print(**f"Error: {**e**}"**)  
  
 **else**:  
 *# disconnect from the SMTP server* smtp\_server.quit()  
  
  
**def** \_handle\_change\_email(main\_server, db, client\_ip, code\_dic, username, email, code):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary with the code sent to the email and the mail as key* **:param** *username: the username of the client* **:param** *email: the email of the client* **:param** *code: the code entered* **:return***: changes the email in the database and sends response to the client  
 """* status = 1  
 **if** email **in** code\_dic:  
 **if** code\_dic[email] == code:  
 status = db.change\_email(username, email)  
 **del** code\_dic[email]  
  
 msg = serverProtocol.pack\_change\_email\_response(status, email)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_change\_password(main\_server, db, client\_ip, username, old\_pass, new\_pass, confirmed\_pass):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *username: the username of the client* **:param** *old\_pass: the old password* **:param** *new\_pass: the new password entered* **:param** *confirmed\_pass: confirmation of the new password entered* **:return***: changes password and sends response to the client  
 """* status = 1  
 **if** new\_pass == confirmed\_pass **and** db.get\_password(username) == encryption.hash\_msg(old\_pass) **and** len(new\_pass) >= 4:  
 status = db.change\_password(username, encryption.hash\_msg(new\_pass))  
  
 msg = serverProtocol.pack\_change\_password\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_change\_photo(files\_server, client\_ip, username, photo\_data):  
 *"""* **:param** *files\_server: the files server object* **:param** *client\_ip: the clients ip* **:param** *username: the client username* **:param** *photo\_data: the data of the photo* **:return***: changes photo and sends response to the client  
 """* **try**:  
 **with** open(**f"{**Settings.USER\_PROFILE\_PHOTOS**}/{**username**}.png"**, **'wb' if** type(photo\_data) **is** bytes **else 'w'**) **as** f:  
 f.write(photo\_data)  
 **except** Exception **as** e:  
 print(str(e))  
  
 params = (**"04"**, username)  
 files\_server.send\_file(client\_ip, params)  
  
  
**def** \_handle\_delete\_profile\_photo(files\_server, client\_ip, username):  
 *"""* **:param** *files\_server: the files server* **:param** *client\_ip: the clients ip* **:param** *username: the clients username* **:return***: deletes profile photo and sends response to the client  
 """* status = sFileHandler.delete\_file(**f"{**Settings.USER\_PROFILE\_PHOTOS**}/{**username**}.png"**)  
 **if** status == 0:  
 params = (**"04"**, **'d'**)  
 files\_server.send\_file(client\_ip, params)  
  
  
**def** \_handle\_send\_email(main\_server, db, client\_ip, code\_dic, username):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary with email and the code that was sent to it* **:param** *username: the username of the user* **:return***: sends email to the user and sends response to the user  
 """* email = db.get\_email(username)  
 \_send\_email(main\_server, db, client\_ip, code\_dic, email)  
  
 **if** email:  
 status = 0  
 **else**:  
 status = 1  
  
 msg = serverProtocol.pack\_send\_email\_response(status, email)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_check\_email(main\_server, db, client\_ip, code\_dic, email, code):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *code\_dic: dictionary built by email and the code that was sent to it* **:param** *email: the clients email* **:param** *code: the code entered* **:return***: checks if the code entered is the same one that was send and sends response to the client  
 """* status = 1  
  
 **if** email **in** code\_dic:  
 **if** code\_dic[email] == code:  
 status = 0  
  
 msg = serverProtocol.pack\_check\_code\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_forgot\_password(main\_server, db, client\_ip, username, password, confirmed\_password):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *username: the username of the client* **:param** *password: the password of the client* **:param** *confirmed\_password: the confirmation of the password* **:return***: changed the password and sends response to the client  
 """* status = 1  
  
 **if** password == confirmed\_password **and** len(password) >= 4:  
 status = db.change\_password(username, encryption.hash\_msg(password))  
  
 msg = serverProtocol.pack\_forgot\_password\_response(status)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_zip\_folder(main\_server, db, client\_ip, folder\_path):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *folder\_path: the path of the folder* **:return***: zips the folder and sends response to the client  
 """* status, folder\_name = sFileHandler.create\_zip(**f"{**Settings.SERVER\_FILES\_PATH**}/{**folder\_path**}"**)  
 msg = serverProtocol.pack\_zip\_folder\_response(status, folder\_name)  
 main\_server.send(client\_ip, msg)  
  
  
**def** \_handle\_disconnect(main\_server, db, client\_ip, users\_by\_ip):  
 *"""* **:param** *main\_server: the server object* **:param** *db: the database* **:param** *client\_ip: the clients ip* **:param** *users\_by\_ip: the dict containing each user by the ip it connected on* **:return***:  
 """* **if** client\_ip **in** users\_by\_ip:  
 **del** users\_by\_ip[client\_ip]  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main\_loop()

serverProtocol:  
**import** os  
**from** settings **import** CurrentSettings **as** Settings  
  
  
**def** unpack\_message(msg):  
 *"""* **:param** *msg: message to unpack* **:return***: tuple with opcode and variables  
 """* opcode = msg[:2]  
 data\_from\_msg = []  
 msg = msg[2:]  
  
 **while** len(msg) > 0:  
 **try**:  
 data, msg = get\_data\_from\_string(msg, 2)  
 data\_from\_msg.append(data)  
  
 **except** Exception **as** e:  
 print(**"in unpack\_message -"**, str(e))  
 opcode = **None  
 break  
  
 return** opcode, data\_from\_msg  
  
  
**def** get\_data\_from\_string(line, leng):  
 *"""* **:param** *line: line to slice* **:param** *leng: length to slice* **:return***: message built by protocol  
 """  
 # getting the data from the length of the string* data = line[leng:leng + int(line[:leng])]  
  
 *# substring* line = line[leng + int(line[:leng])::]  
  
 **return** data, line  
  
  
**def** pack\_files\_message(username):  
 *"""* **:param** *username: username of user* **:return***: string with all files and directories of user in the drive  
 """* files\_of\_user = **"03"** path = **f"{**Settings.SERVER\_FILES\_PATH**}/{**username**}"  
  
 for** (dirname, dirs, files) **in** os.walk(path):  
 *# removing the path so that it's only the directories in the system and not from the server* dirname = dirname[len(path) + 1::].replace(**'\\'**, **'/'**)  
  
 *# adding the length of the name and the name* files\_of\_user += str(len(dirname)).zfill(2) + dirname  
  
 *# adding the length of the directories and the directories list* dirs = **'!'**.join(dirs)  
 files\_of\_user += str(len(dirs)).zfill(4) + dirs  
  
 *# adding the length of the files and the files list* files = **'@'**.join(files)  
 files\_of\_user += str(len(files)).zfill(4) + files  
  
 files\_of\_user += **'\n'  
  
 return** files\_of\_user  
  
  
**def** pack\_register\_response(response, username, password, email):  
 *"""* **:param** *response: response to register* **:param** *username: username of user* **:param** *password: password of user* **:param** *email: email of user* **:return***: message built by protocol  
 """* **return** (**f"01{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(username))).zfill(2)**}{**username**}"  
 f"{**str(len(str(password))).zfill(2)**}{**password**}{**str(len(str(email))).zfill(2)**}{**email**}"**)  
  
  
**def** pack\_login\_response(response, email):  
 *"""* **:param** *response: response to login* **:return***: message built by protocol  
 """* **return f"02{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(email))).zfill(2)**}{**email**}"  
  
  
def** pack\_verify\_check\_response(response):  
 *"""* **:param** *response: response to verify check* **:return***: message built by protocol  
 """* **return f"15{**str(len(str(response))).zfill(2)**}{**response**}"  
  
  
def** pack\_file\_download\_response(response, datalen, path, selected\_path):  
 *"""* **:param** *response: response to download file* **:param** *datalen: length of the data* **:param** *path: path of the file they asked to download* **:param** *selected\_path: path of the user where to download* **:return***: message built by protocol  
 """* **return f"11{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(datalen))).zfill(2)**}{**datalen**}"** \  
 **f"{**str(len(str(path))).zfill(2)**}{**path**}{**str(len(str(selected\_path))).zfill(2)**}{**selected\_path**}"  
  
  
def** pack\_upload\_port(port):  
 *"""* **:param** *port: upload port* **:return***: message built by protocol  
 """* **return f"16{**str(len(str(port))).zfill(2)**}{**port**}"  
  
  
def** pack\_file\_upload\_response(response, path):  
 *"""* **:param** *response: response to file upload* **:param** *path: path of file* **:return***: message built by protocol  
 """* **return f"12{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(path))).zfill(2)**}{**path**}"  
  
  
def** pack\_create\_folder\_response(response):  
 *"""* **:param** *response: response to folder create* **:return***: message built by protocol  
 """* **return f"13{**str(len(str(response))).zfill(2)**}{**response**}"  
  
  
def** pack\_delete\_response(response):  
 *"""* **:param** *response: response to file delete* **:param** *path: the path deleted* **:return***: message built by protocol  
 """* **return f"10{**str(len(str(response))).zfill(2)**}{**response**}"  
  
  
def** pack\_rename\_file\_response(response, new\_name):  
 *"""* **:param** *response: response to file delete* **:param** *path: the path deleted* **:param** *new\_name: the name of the file* **:return***: message built by protocol  
 """* **return f"08{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(new\_name))).zfill(2)**}{**new\_name**}"  
  
  
def** pack\_share\_response(response):  
 *"""* **:param** *response: response to share file* **:return***: message built by protocol  
 """* **return f"09{**str(len(str(response))).zfill(2)**}{**response**}"  
  
  
def** pack\_change\_photo\_response(response, file\_len):  
 *"""* **:param** *response: response to change photo* **:return***: message built by protocol  
 """* **return f"04{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(file\_len))).zfill(2)**}{**file\_len**}"  
  
  
def** pack\_change\_password\_response(response):  
 *"""* **:param** *response: response to change password* **:return***: message built by protocol  
 """* **return f"06{**str(len(str(response))).zfill(2)**}{**response**}"  
  
  
def** pack\_change\_email\_response(response, email):  
 *"""* **:param** *response: response to change email* **:param** *email: email to change to* **:return***: message built by protocol  
 """* **return f"05{**str(len(str(response))).zfill(2)**}{**response**}{**str(len(str(email))).zfill(2)**}{**email**}"  
  
  
def** pack\_login\_verify\_response(status):  
 *"""* **:param** *status: response to login verify* **:return***: message built by protocol  
 """* **return f"23{**str(len(str(status))).zfill(2)**}{**status**}"  
  
  
def** pack\_add\_shared\_file(path):  
 *"""* **:param** *path: path of file to add to client branches* **:return***: message built by protocol  
 """* **return f"14{**str(len(str(path))).zfill(2)**}{**path**}"  
  
  
def** pack\_send\_email\_response(status, email):  
 **return f"17{**str(len(str(status))).zfill(2)**}{**status**}{**str(len(str(email))).zfill(2)**}{**email**}"  
  
  
def** pack\_move\_file\_response(status, new\_folder):  
 *"""* **:param** *status: response to move file* **:param** *new\_folder: the folder moved to* **:return***: message built by protocol  
 """* **return f"18{**str(len(str(status))).zfill(2)**}{**status**}{**str(len(str(new\_folder))).zfill(2)**}{**new\_folder**}"  
  
  
def** pack\_paste\_file\_response(status):  
 *"""* **:param** *status: response to paste file* **:return***: message built by protocol  
 """* **return f"19{**str(len(str(status))).zfill(2)**}{**status**}"  
  
  
def** pack\_open\_file\_response(status, file\_path, data\_len):  
 *"""* **:param** *status: status to open file* **:param** *file\_path: the file path in the server* **:param** *data\_len: the length of the data of the file* **:return***: message built by protocol  
 """* **return f"20{**str(len(str(status))).zfill(2)**}{**status**}{**str(len(str(data\_len))).zfill(2)**}{**data\_len**}"** \  
 **f"{**str(len(str(file\_path))).zfill(2)**}{**file\_path**}"  
  
  
def** pack\_get\_details\_response(status, email, data\_len):  
 *"""* **:param** *status: response to get details* **:param** *email: email of the client* **:param** *data\_len: length of the profile photo* **:return***: message built by protocol  
 """* **return** (**f"21{**str(len(str(status))).zfill(2)**}{**status**}"  
 f"{**str(len(str(email))).zfill(2)**}{**email**}{**str(len(str(data\_len))).zfill(2)**}{**data\_len**}"**)  
  
  
**def** pack\_delete\_profile\_photo\_response(status):  
 *"""* **:param** *status: response to delete photo request* **:return***: message built by protocol  
 """* **return f"22{**str(len(str(status))).zfill(2)**}{**status**}"  
  
  
def** pack\_register\_verify\_response(status):  
 *"""* **:param** *status: response to register verify* **:return***: message built by protocol  
 """* **return f"24{**str(len(str(status))).zfill(2)**}{**status**}"  
  
  
def** pack\_check\_code\_response(status):  
 *"""* **:param** *status: response to check code* **:return***: message built by protocol  
 """* **return f"25{**str(len(str(status))).zfill(2)**}{**status**}"  
  
  
def** pack\_forgot\_password\_response(status):  
 *"""* **:param** *status: response to forgot password request* **:return***: message built by protocol  
 """* **return f"26{**str(len(str(status))).zfill(2)**}{**status**}"  
  
  
def** pack\_zip\_folder\_response(status, zipped\_folder\_name):  
 *"""* **:param** *status: response to zip folder* **:param** *zipped\_folder\_name: zipped folder name* **:return***: message built by protocol  
 """* **return f"27{**str(len(str(status))).zfill(2)**}{**status**}{**str(len(str(zipped\_folder\_name))).zfill(2)**}{**zipped\_folder\_name**}"  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 code, data\_from\_message = unpack\_message(**"1210reef/reef1000012"**)  
 dataa = pack\_files\_message(**"reef"**)  
  
 print(dataa)

sFileHandler:  
**import** os  
**import** zipfile  
  
  
**def** rename\_file(path, new\_name):  
 *"""* **:param** *path: path of file to rename* **:param** *new\_name: the new name of the file* **:return***: renames the file and returns 1 if it didn't work and 0 if it manged  
 """* status = 1  
 **if** os.path.exists(path):  
 **try**:  
 os.path.dirname(path)  
 os.rename(path, os.path.dirname(path) + **'/'** + new\_name)  
 status = 0  
 **except** Exception **as** e:  
 print(str(e))  
 **else**:  
 print(**"Isn't a file"**)  
  
 **return** status  
  
  
**def** delete\_file(path):  
 *"""* **:param** *path: path of file to delete* **:return***: deletes file 1 if successful 0 if failed  
 """* status = 0  
 **if** os.path.exists(path):  
 **try**:  
 os.remove(path)  
  
 print(**"Deleted file at - "** + path)  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
 **else**:  
 status = 2  
  
 **return** status  
  
  
**def** generate\_unique\_zip\_name(path, base\_name):  
 *"""* **:param** *path: path of the folder to zip* **:param** *base\_name: the name of the folder* **:return***: creates the name based on the existing files  
 """* full\_path = **f"{**path**}/{**base\_name**}.zip"** unique\_name = **f"{**base\_name**}.zip"  
  
 if** os.path.exists(full\_path):  
 index = 1  
 **while** os.path.exists(**f"{**path**}/{**base\_name**} ({**index**}).zip"**):  
 index += 1  
 unique\_name = **f"{**base\_name**} ({**index**}).zip"  
  
 return** unique\_name  
  
  
**def** create\_zip(total\_path):  
 *"""* **:param** *total\_path: path of the folder* **:return***: zips the path returns 1 if failed to zip 0 if it worked  
 """* status = 0  
 zip\_file\_name = **None  
 if** os.path.isdir(total\_path):  
 **try**:  
 folder\_path, name = total\_path.split(**'/'**)[:-1], total\_path.split(**'/'**)[-1]  
 folder\_path = **'/'**.join(folder\_path)  
  
 *# generates the correct name* zip\_file\_name = generate\_unique\_zip\_name(folder\_path, name)  
 zip\_path = **f"{**folder\_path**}/{**zip\_file\_name**}"** *# Create a ZipFile object in write mode* **with** zipfile.ZipFile(zip\_path, **'w'**) **as** zipf:  
 *# Add all files in the directory and its subdirectories to the zip file* **for** root, \_, files **in** os.walk(total\_path):  
 **for** file **in** files:  
 file\_path = **f"{**root**}/{**file**}"** file\_name = os.path.relpath(file\_path, total\_path)  
 zipf.write(file\_path, arcname=file\_name)  
  
 **except** Exception **as** e:  
 print(str(e))  
 status = 1  
 **else**:  
 status = 1  
  
 **return** status, zip\_file\_name  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 create\_zip(**"C:/Users/reefg/PycharmProjects/goldrive\_code/ServerFiles"**)

encryption:  
**import** base64  
**from** Cryptodome.Cipher **import** AES  
**from** Cryptodome **import** Random  
**import** hashlib  
**import** secrets  
**import** chardet  
  
  
**class** Encryption(object):  
 **def** \_\_init\_\_(self, key):  
 *"""* **:param** *key: the encryption key  
 """* self.bs = AES.block\_size  
 self.key = hashlib.sha256(str(key).encode()).digest()  
  
 **def** enc\_msg(self, message):  
 *"""* **:param** *message: the message to encrypt* **:return***: the encrypted message  
 """* **if** type(message) == str:  
 message = message.encode()  
 raw = self.\_pad(message)  
 iv = Random.new().read(AES.block\_size)  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
 **return** base64.b64encode(iv + cipher.encrypt(raw))  
  
 **def** dec\_msg(self, encrypt\_message):  
 *"""* **:param** *encrypt\_message: the encrypted message to decrypt* **:return***: the decrypted message  
 """* enc = base64.b64decode(encrypt\_message)  
 iv = enc[:AES.block\_size]  
 cipher = AES.new(self.key, AES.MODE\_CBC,iv)  
 a = self.\_unpad(cipher.decrypt(enc[AES.block\_size:]))  
 encoding = chardet.detect(a)  
 **if** encoding[**'encoding'**] == **"ascii"**:  
 a = a.decode()  
 **return** a  
  
 **def** \_pad(self, message):  
 *"""* **:param** *message: message* **:return***: the message with the padding  
 """* **return** message + (self.bs - len(message) % self.bs) \* chr(self.bs - len(message) % self.bs).encode()  
  
 @staticmethod  
 **def** \_unpad(message):  
 *"""* **:param** *message: encrypted message* **:return***: the encrypted message without the padding  
 """* **return** message[:-ord(message[len(message) - 1:])]  
  
  
**def** hash\_msg(msg):  
 *"""* **:param** *msg: the msg to hash* **:return***: the hash of the msg  
 """* **return** hashlib.sha3\_256(msg.encode()).hexdigest()  
  
  
**def** get\_dh\_factor():  
 *"""* **:return***: creates the dh factor  
 """* private\_key = secrets.SystemRandom().randint(1, p)  
  
 **return** private\_key, (g \*\* private\_key) % p  
  
  
**def** create\_symmetry\_key(private\_key, shared\_key):  
 *"""* **:param** *private\_key: the personal key* **:param** *shared\_key: the key i got from the server* **:return***: returns the symmetrical encryption object from the dh key change  
 """* **return** Encryption((shared\_key \*\* private\_key) % p)  
  
  
p = 7723  
g = 1229  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 a, A = get\_dh\_factor()  
 print(len(str(A)))  
 b, B = get\_dh\_factor()  
  
 keyServer = create\_symmetry\_key(a, B)  
 keyClient = create\_symmetry\_key(b, A)  
 print(keyClient.key)  
 print(keyServer.key)  
  
 **with** open(**r"T:\public\יב\imri\projectCode\files\cat.jpg"**, **'rb'**) **as** f:  
 data = f.read()  
  
 print(len(data))  
  
 encM = keyServer.enc\_msg(data)  
 print(len(encM), type(encM))  
  
 decM = keyClient.dec\_msg(encM)  
 print(decM)

monitorFile:  
**import** win32file  
**import** subprocess  
**import** queue  
**import** psutil  
  
FILE\_LIST\_DIRECTORY = 0x0001  
FILE\_NOTIFY\_CHANGE\_FILE\_NAME = 0x0001  
FILE\_NOTIFY\_CHANGE\_DIR\_NAME = 0x0002  
FILE\_NOTIFY\_CHANGE\_LAST\_WRITE = 0x0010  
FILE\_FLAG\_BACKUP\_SEMANTICS = 0x02000000  
OPEN\_EXISTING = 3  
  
*# the action name for the change*file\_actions = {  
 0x00000001:  
 **"Added"**,  
 0x00000002:  
 **"Removed"**,  
 0x00000003:  
 **"Modified"**,  
 0x00000004:  
 **"Renamed old name"**,  
 0x00000005:  
 **"Renamed new name"**}  
image\_types = [**"apng"**, **"avif"**, **"gif"**, **"jpg"**, **"jpeg"**, **"jfif"**, **"pjpeg"**, **"pjp"**, **"png"**, **"svg"**, **"webp"**, **"bmp"**, **"ico"**, **"cur"**,  
 **"tif"**, **"tiff"**]  
  
*# the default apps to open for each type*default\_for\_type = {**'txt'**: **'Notepad.exe'**,  
 **'docx'**: **'WINWORD.EXE'**,  
 **'doc'**: **'WINWORD.EXE'**,  
 **'pptx'**: **'POWERPNT.EXE'**,  
 **'ppt'**: **'POWERPNT.EXE'**,  
 **'zip'**: **'explorer.exe'**,  
 \*\*{img: **'Microsoft.Photos.exe' for** img **in** image\_types},  
 **'xlsx'**: **'EXCEL.EXE'**,  
 **'xls'**: **'EXCEL.EXE'**}  
  
  
**def** monitor(path\_to\_watch, q):  
 *"""* **:param** *path\_to\_watch: path of directory to watch changes made in* **:param** *q: queue to update info in* **:return***: checks for changes and puts Changed in the queue when a change is made in the directory given  
 """* 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**:  
 **while True**:  
 **try**:  
 result = win32file.ReadDirectoryChangesW(  
 directory\_handle,  
 4096,  
 **True**, *# Watch subtree* FILE\_NOTIFY\_CHANGE\_LAST\_WRITE | FILE\_NOTIFY\_CHANGE\_FILE\_NAME,  
 **None** )  
  
 **for** action **in** result:  
 **if** file\_actions[action[0]] == **"Modified" and not** action[1].startswith(**'~'**):  
 q.put(**"Changed"**)  
 **except** Exception **as** e:  
 **break  
  
  
def** get\_all\_pid(process\_name):  
 *"""* **:param** *process\_name: get all pid for a process* **:return***: the current pid open for the process checking  
 """* current = []  
  
 **for** p **in** psutil.process\_iter():  
 **if** p.name() == process\_name:  
 current.append(p.pid)  
  
 **return** current  
  
  
**def** wait\_until(file\_path, q):  
 *"""* **:param** *file\_path: file path* **:param** *q: queue to put changes in* **:return***: opens the file and when the user closes it puts in the queue finished  
 """* 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'** ls1 = get\_all\_pid(process\_name)  
 subprocess.Popen([**'start'**, file\_path], shell=**True**)  
  
 **while True**:  
 ls2 = get\_all\_pid(process\_name)  
 **if** ls2 != ls1:  
 **break** new\_pid = set(ls2) - set(ls1)  
 pid = list(new\_pid)[0]  
  
 **while** psutil.pid\_exists(pid):  
 **pass** q.put(**"Finished"**)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 Q = queue.Queue()  
  
 *# file\_path = r"T:\public\reefcheck\check.docx"  
 # 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'  
 #  
 # print(process\_name)  
 #  
 # wait\_until(process\_name, file\_path, q)*

settings:  
**class** ClassSettings:  
 **def** \_\_init\_\_(self):  
 self.SERVERPORT = 1234  
 self.SERVERIP = **"192.168.4.75"** self.USER\_FILES\_PATH = **r"D:\!ReefGold\goldrive\_code\UserGraphics"**.replace(**"\\"**, **'/'**)  
 self.SERVER\_FILES\_PATH = **r"D:\!ReefGold\users\_files"**.replace(**"\\"**, **'/'**)  
 self.USER\_PROFILE\_PHOTOS = **r"D:\!ReefGold\user\_profile\_photos"**.replace(**'\\'**, **'/'**)  
  
  
**class** HomeSettings:  
 **def** \_\_init\_\_(self):  
 self.SERVERPORT = 1234  
 self.SERVERIP = **"10.0.0.12"** self.USER\_FILES\_PATH = **r"C:\Users\reefg\PycharmProjects\goldrive\_code\UserGraphics"**.replace(**"\\"**, **'/'**)  
 self.SERVER\_FILES\_PATH = **r"C:\Users\reefg\user\_files"**.replace(**"\\"**, **'/'**)  
 self.USER\_PROFILE\_PHOTOS = **r"C:\Users\reefg\user\_profile\_photos"**.replace(**'\\'**, **'/'**)  
  
  
**class** CurrentSettings:  
 settings = ClassSettings()  
 SERVERPORT = settings.SERVERPORT  
 SERVERIP = settings.SERVERIP  
 USER\_FILES\_PATH = settings.USER\_FILES\_PATH  
 SERVER\_FILES\_PATH = settings.SERVER\_FILES\_PATH  
 USER\_PROFILE\_PHOTOS = settings.USER\_PROFILE\_PHOTOS