ה א ו נ י ב ר ס י ט ה ה פ ת ו ח ה

ה ח ט י ב ה ל מ ד ע י ה מ ח ש ב

ת ו א ר ר א ש ו ן



סדנה בתקשורת מחשבים (20588).

# תיק פרויקט

## עבור

## תיקייה מסונכרנת

מגיש:  
ברק שמאילוב, ת"ז: 205494628.

מנחה:  
דני כלפון.

סמסטר:

קיץ 23ג.

תוכן עניינים

1. **מבוא 3**
2. **דרישות קדם 3**
3. **הוראות התקנה 3**
4. **הוראות שימוש 4**
5. **למידת רקע** **13**
6. **ארכיטקטורה 14**
7. **בדיקות וטסטים 34**
8. **פערים במצב הקיים 36**

1. מבוא:

תוכנת הפרויקט הינה תיקייה מסונכרנת, אופן המימוש הוא באמצעות שרת כגון: Dropbox, Google Drive.

התוכנה מאפשרת חיבור של מספר משתמשים על גבי הרשת המקומית, לאחר שמשתמש מתחבר לתיקייה הוא רואה את כל הקבצים שנמצאים בשרת, הקבצים מוצגים למשתמש באמצעות ממשק משתמש שמדמה תיקייה.

יכולות התוכנה הן:

- העלאה של קבצים/תיקייה עם קבצים לשרת.

- יצירת תיקיות חדשות.

-הורדה של קבצים לפי קובץ או מספר קבצים מהשרת.

- שינוי שם של הקבצים/תיקיות בשרת.

- מחיקת הקבצים/תיקיות בשרת.

- מעבר בין תתי תיקיות לתיקיות אם.

- העברת תיקיות וקבצים מתיקייה לתיקייה.

2. דרישות קדם:

התוכנה נכתבה בשפת פייתון בגרסה 3.9.7, התוכנה נבדקה במערכת הפעלה ווינדוס /1110.

3. הוראות התקנה:

1. תחילה יש ליצור פרויקט חדש ב IDE (הIDE שעבדתי איתו הוא PyCharm) לאחר מכן יש להעתיק את תוכן התיקייה Project Source Files שנמצאת בZIP אל הפרויקט החדש שפתחנו בIDE.

2. כעת יש להתקין ספריה חיצונית אחת בשם PIL באמצעות הטרמינל בעזרת הפקודה –

pip install pillow.

3. לאחר ההתקנה של הספרייה, אפשר להפעיל את את הקובץ server.py שמפעיל את השרת.

ולאחר מכן גם את האפליקציה client.py.

4. הוראות שימוש:

שרת:

לאחר שנפעיל את השרת יעלה לנו מסך הפתיחה שנראה כך:

תמונה שמכילה טקסט, צילום מסך, תצוגה, תוכנה

התיאור נוצר באופן אוטומטי

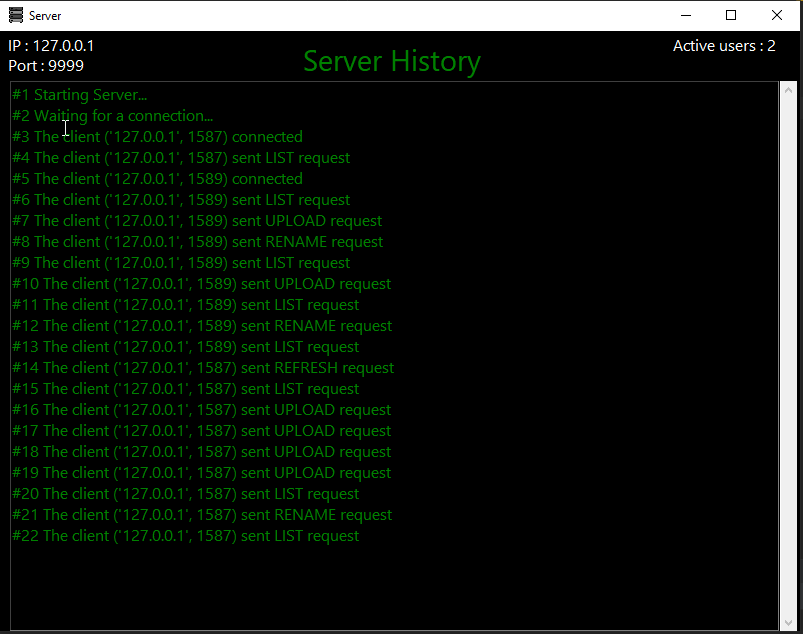
ישנה אופציה ללחוץ על הריבוע הקטן שליד התיבת טקסט על מנת להשתמש בPORT או הIP שהם הברירת מחדל. ברגע שנסמן לא ניתן יהיה להכניס ערכים לתיבת הטקסט.

דוגמה:



אם לא בחרנו באופציה של הברירת מחדל נצטרך לרשום את הפרטי הIP והPORT בתיבת הטקסט ולאחר שרשמנו את הפרטים נלחץ על הכפתור "setup server".

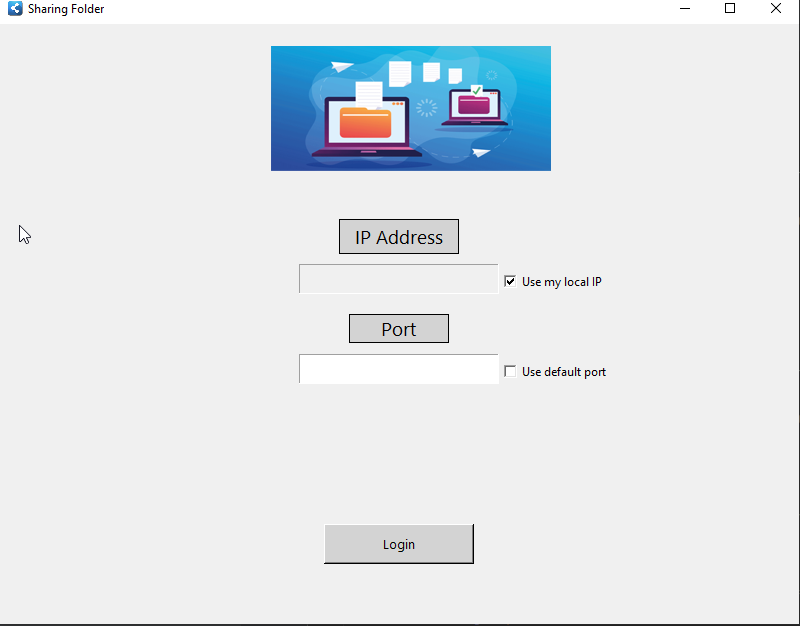
לאחר שלחצנו יעלה המסך הבא:



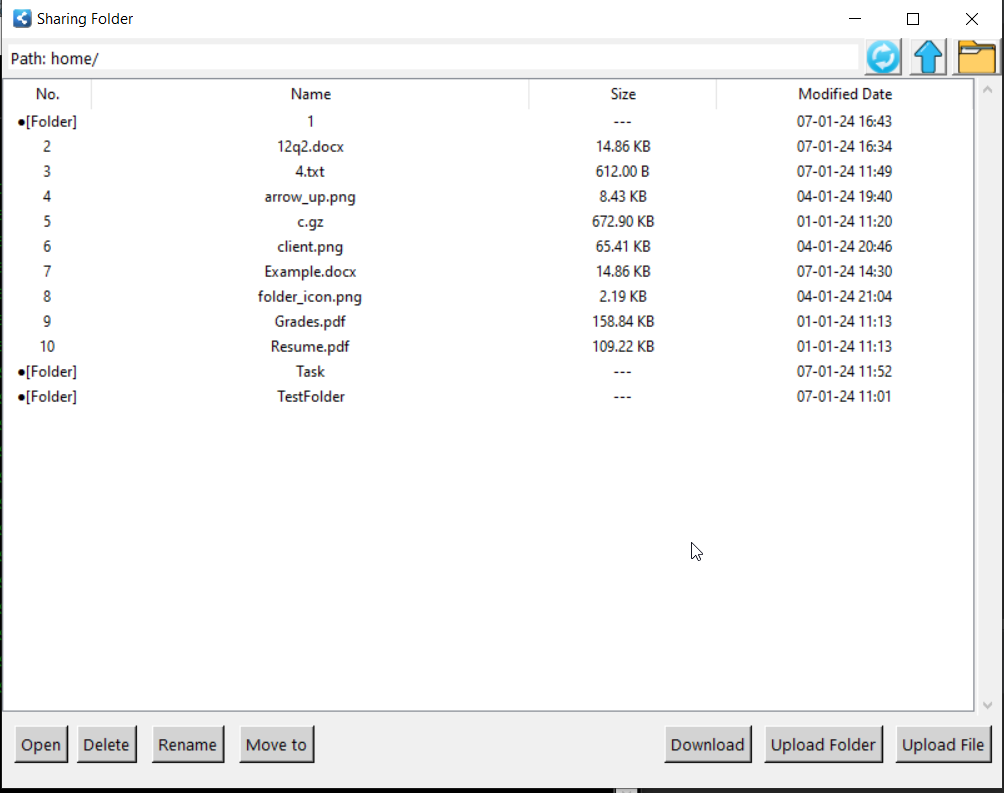
* כפי שנוכל לראות מצד שמאל הכי למעלה נוכל לראות את כתובת הIP של השרת והPORT של השרת.
* בצד ימין הכי למעלה נוכל לראות מספר המשתמשים המחוברים לשרת.
* בגוף של החלון נוכל לראות את ההיסטוריה של השרת, כלומר מי התחבר, מי התנתק, איזה בקשה שלח כל משתמש לשרת וכן הלאה (מידע על סוגי הבקשת יפורט בהמשך).
* בצד הכי ימני של החלון ניתן לראות סרגל צד (Scroll Bar) שניתן לגרירה כאשר החלון מתמלא בהודעות.
* לחיצה על הכפתור X בחלון תגרום לסגירת השרת וליציאה מהתוכנית.

לקוח:

לאחר שנפעיל נקבל את החלון הבא:



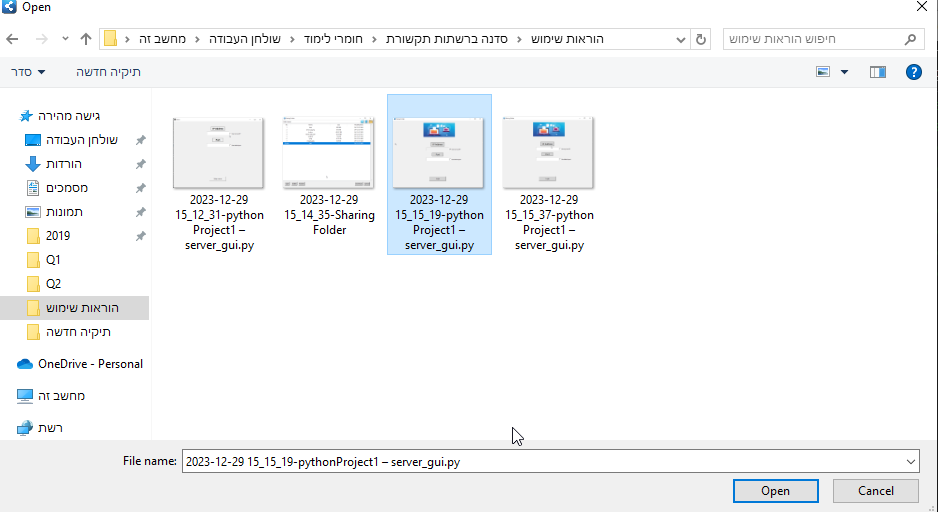
ההסבר דומה לשרת, בוחרים את כתובת הIP והPORT של השרת שאליו אנחנו רוצים להתחבר גם פה יש אופציה של בחירת ברירת המחדל, לאחר שבחרנו נלחץ על הכפתור Login ויפתח לנו המסך הבא (בתנאי שהשרת הוקם וקיים כבר):



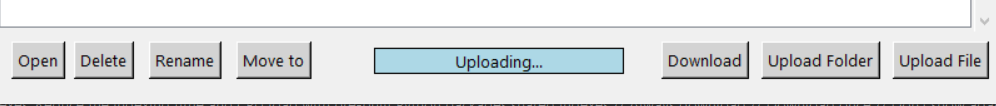
* במסך נוכל לראות את הקבצים בדומה לאיך שהם מוצגים בתיקייה במחשב, כל קובץ מקבל מספור משלו כפי שניתן לראות בעמודה No. אם מדובר בתיקייה ולא בקובץ יהיה רשום ●[Folder], שם הקובץ בעמודה Name, גודל הקובץ בעמודה Size ובעמודה האחרונה Modified Date את התאריך של השינוי האחרון.
* כפי שניתן לראות בחלק העליון יוצג לנו הנתיב שבו אנחנו נמצאים כרגע אנחנו בתיקיית הבית home.
* מצד ימין לנתיב ישנו כפתור של שני חצים בצורת מעגל, כפתור זה כפתור "רענון", כלומר ברגע שנלחץ על הכפתור תשלח בקשה לשרת שיציג לנו את הקבצים והתיקיות שנמצאים כרגע אצלו, כפתור זה נועד במקרה שאם משתמש אחר עושה שינוי, משתמש שני שמחובר לשרת אינו מודע לזה וברגע שילחץ על הכפתור יוצגו השינויים שעשה המשתמש הראשון.
* הכפתור הבא שנמצא ליד כפתור ה"רענון" הוא הכפתור עם החץ למעלה, כפתור זה נועד במקרה שאם אנחנו לא בתיקיית הבית home אלא נמצאים בתת תיקייה נוכל לעלות בנתיב למעלה לתיקיית הבית home.
* הכפתור הבא שליד הכפתור עם החץ למעלה זה כפתור עם לוגו של תיקייה, כפתור זה נותן לנו את האפשרות ליצור תיקיות חדשות/תתי תיקיות.
* על מנת לצאת מהאפליקציה נעשה זאת דרך הכפתור X בחלון.
* שבעת הכפתורים שלמטה משמשים לפעולות על הקבצים ועל התיקיות, נסביר על כל אחד מהכפתורים, הראשון:

Upload File :

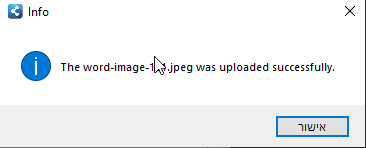
ברגע שנלחץ על הכפתור יפתח לנו החלון הבא, שבמסגרתו נצטרך לבחור את הקובץ שברצוננו לעלות, לאחר שבחרנו נלחץ על open ותהליך העלאה יכנס לעבודה.



בזמן העלאה יוצג לנו בתחתית החלון בין הכפתורים סטטוס העלאה:



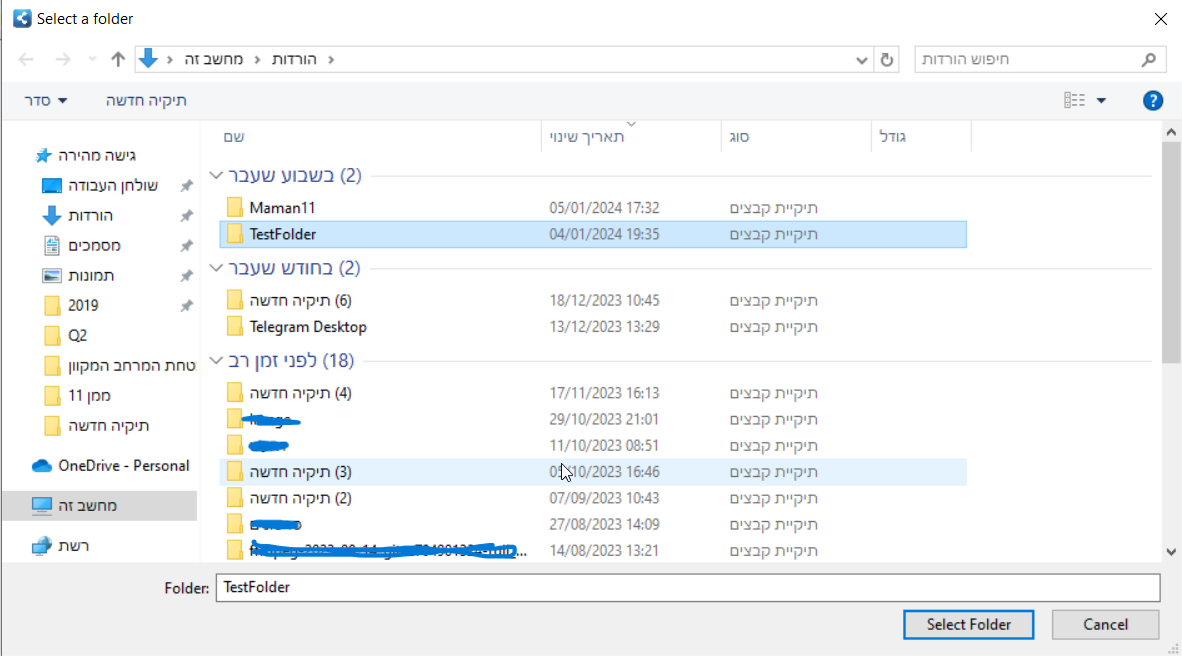
לאחר שההעלאה תסתיים תקפוץ לנו הודעה שתגיד לנו אם ההעלאה הצליחה או שהייתה שגיאה:



לאחר מכן הקובץ החדש שהעלאנו יוצג לנו באופן אוטומטי החלון.

Upload Folder:

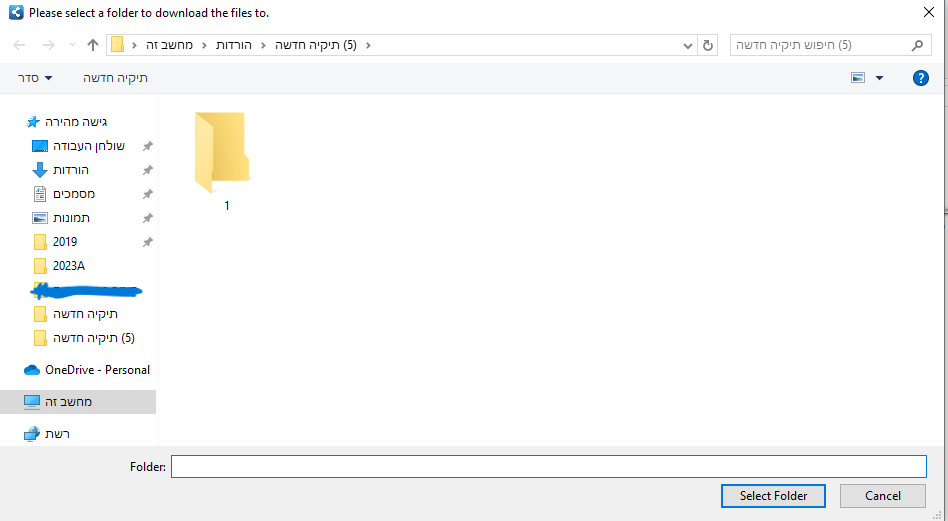
לאחר שנלחץ על הכפתור יפתח לנו החלון הבא ובו נוכל לבחור את התיקייה שאנו רוצים לעלות (אפשר לבחור רק תיקיות):



לאחר שנבחר את התיקייה נלחץ על הכפתור "select folder", תהליך העלאת התיקייה יכנס לביצוע.

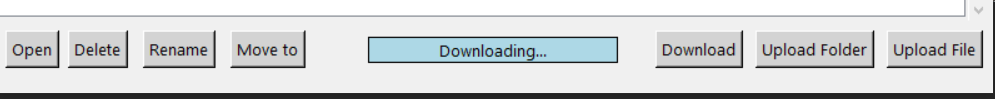
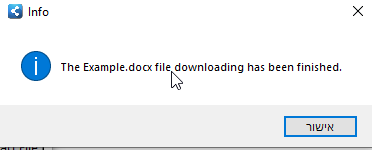
Download:

תחילה נסמן את הקובץ או הקבצים שאותם אנו רוצים להוריד מהשרת ונלחץ על הכפתור download, ברגע שנלחץ יפתח לנו החלון הבא:



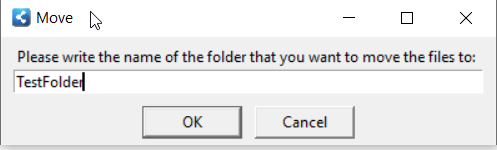


נוכל לבחור לאיפה נרצה להוריד למחשב שלנו באמצעות החלון, נשים לב שנצטרך לבחור תיקייה להוריד לתוכה את כל הקבצים שסימנו לאחר שבחרנו נלחץ על הכפתור Select Folder, אם שורת הטקסט Folder: שמעליה הקו הצהוב אין בה טקסט אזי התיקייה שבה יורדו הקבצים יהיו בתיקייה שכרגע אנו נמצאים.

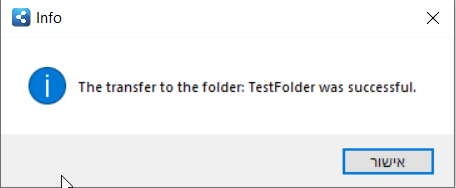
לאחר מכן תעלה לנו הודעה שההורדה אכן הצליחה עבור כל קובץ, וגם יוצג לנו סטטוס ההורדה:  


Move to:

תחילה נסמן את הקובץ ותיקייה שאנו רוצים להעביר, נוכל להעביר לתתי תיקיות שבתיקייה שאנו נמצאים או בתיקיות שמעל התיקייה שאנו נמצאים בנתיב, על מנת להעביר לתיקייה נלחץ על הכפתור move to לאחר שסימנו את הקובץ/התיקייה שאנו רוצים להעביר, יפתח לנו החלון הבא:



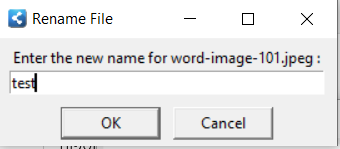
ונרשום את שם התיקייה, לדוגמה בחרנו להעביר קובץ לתיקייה TestFolder, לאחר שההעברה תסתיים תוצג לנו ההודעה הבאה:



נוכל להעביר גם בנוסף לתיקיות שמעל התיקייה שאנו נמצאים בנתיב לדוגמה אם הנתיב שבו אנחנו נמצאים הוא home/TestFolder/NewFolder/ אז נוכל להעביר קבצים/תיקיות לTestFolder או לhome.

Rename:

תחילה נצטרך לסמן את הקובץ/תיקייה שאנו רוצים לשנות את השם, לאחר שבחרנו נלחץ על הכפתור rename ויקפוץ לנו החלון הקטן הבא:



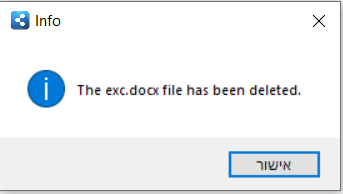
בחלון נוכל לרשום את השם שאנו רוצים, בדוגמה רשמתי test, לאחר מכן נלחץ על OK ותקפוץ לנו ההודעה הבאה שהשינוי הצליח:

תמונה שמכילה טקסט, גופן, צילום מסך

התיאור נוצר באופן אוטומטי

Delete:

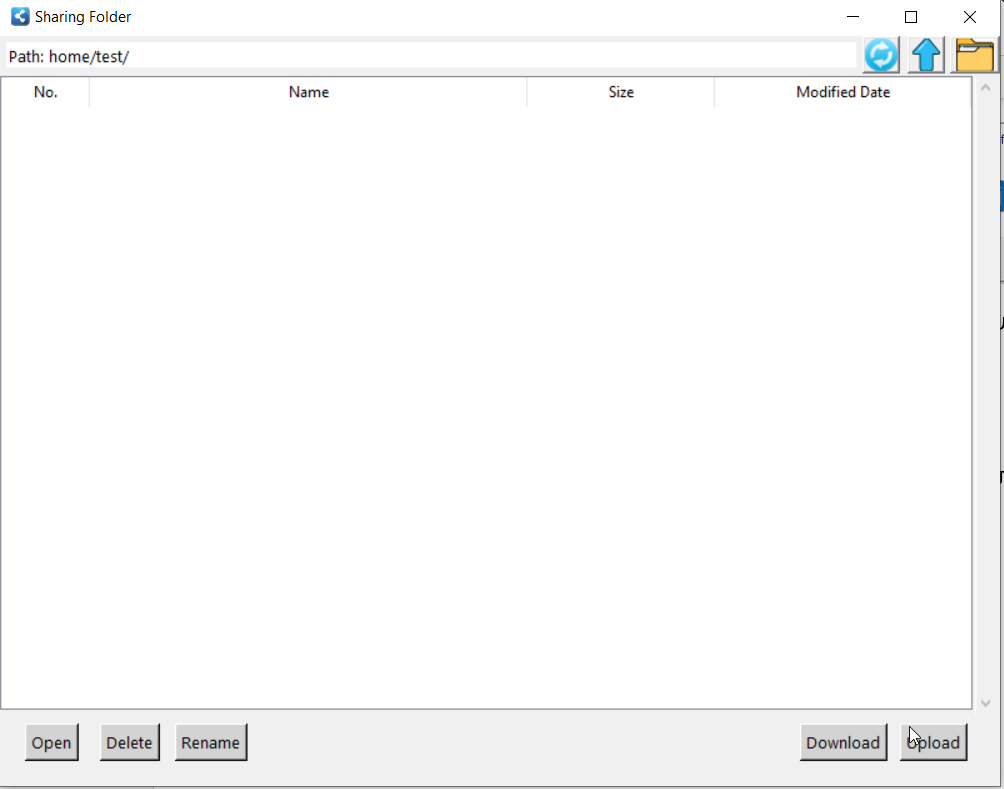
תחילה נצטרך לסמן את הקובץ/תיקייה שאנו רוצים למחוק, לאחר שבחרנו נלחץ על הכפתור delete ,יקפוץ לנו החלון הקטן הבא:



שיאשר לנו שהמחיקה בוצעה בהצלחה.

Open:

תחילה נצטרך לסמן את התיקייה שאנו רוצים לפתוח (לא תקף על קבצים), לאחר שבחרנו נלחץ על הכפתור open ונכנס לתת התיקייה:



כפי שניתן לראות נכנס לתת התיקייה test והנתיב המעודכן מוצג לנו למעלה, התיקייה ריקה ולכן אין קבצים.

5. למידת רקע:

במטרה לממש את הפרויקט רציתי לממש את הפרויקט כאפליקצית DESKTOP המורכבת מצד שרת ומצד לקוח, והממשק שיהיה לה יהיה דומה לתיקייה כמו בווינדוס.

1. אחת הבעיות הייתה לממש לפי הנראות שרציתי, לאחר חקירה באינטרנט, הבנתי שאת צד ממשק הגרפי של הלקוח אני יכול לממש באמצעות ספריית Tkinter שאיתה אפשר לממש את נראות התיקייה באמצעות הwidget tree ו- scroll bar וכך יממוש הממשק הגרפי של הלקוח.
2. לגבי מימוש צד לקוח ושרת, הבעיה הייתה כיצד לממש את התקשורת ואיך להעביר את הנתונים של הקבצים בצורה יעילה שלא תעמיס על הרשת.

הפתרון שבחרתי לאחר בדיקה שלי ברשת, היא מימוש התקשורת באמצעות sockets על גבי רשת TCP שהיא רשת אמינה בשביל העברת קבצים.

הקבצים נקראים באופן בינארי ונשלחים בכל פעם בchunk של 1kb על מנת לא להעמיס על הרשת ושהתעבורה תהיה קלילה וחלקה, בנוסף ישנה בדיקה של הchecksum של הקובץ על מנת לוודא שבאמת הקובץ הגיע כמו שהיה אמור להיות לצד המקבל.

1. בעיה נוספת כיצד נעשה את הפעולות על הקבצים ? כלומר איך נדע לקבל את הנתיב של הקובץ שאותו המשתמש רוצה לעלות או לחלפוין להוריד קובץ מהשרת אל המיקום במחשב האישי שלו ?

הפתרון שמצאתי הוא שימוש באמצעות filedialog זו פונקציה שמגיעה עם הספריה Tkinter ואיתה אפשר לפתוח חלון שבו המשתמש יכול לבחור את הנתיב שהוא רוצה לשמור את הקובץ או לחלופין חלון שבו הוא בוחר קובץ/תיקייה לעלות לשרת.

1. בעיה נוספת איך נעשה פעולות על קבצים בשרת, כגון: שינוי שם, מחיקת קבצים, יצירת תיקייה חדשה, מעבר בין תיקיות וכו'..

הפתרון שבחרתי לאחר הבדיקה הוא באמצעות ספריית os,shutil שמאפשרים לבצע פעולות על קבצים ותיקיות שאנו נדרשים לבצע במסגרת מימוש הפרויקט.

1. בעיה נוספת היא כיצד נתקשר עם השרת והלקוח ונוודא את תקינות הפעולות שהמשתמש מבקש לבצע.

הפתרון שנבחר הוא לכל בקשה/פעולה שהמשתמש מבקש לעשות נעשה מין פרוטוקול ייחודי לכל פעולה למשל העלאת קובץ לשרת תוגדר כבקשה ברגע שהשרת מקבל אותה הוא מבין שהלקוח רוצה לעלות קובץ אליו ולאחר מכן בהתאם לפעולה ולשגיאות שיכולות להיות בעקבותיה ישלחו הודעות מהשרת ללקוח בהתאם, למשל אם הCS של הקובץ שהתקבל בשרת לא תואם אז תשלח הודעת ERROR מהשרת ללקוח ותוקפץ הודעת שגיאה ללקוח שתסביר לו שהקובץ הגיע לא תקין לשרת, ככה נעשה לכל פעולה של המשתמש.

1. בעיה של שימוש במשאב משותף, ישנו מצב שבו משתמש אחד מוריד קובץ ומשתמש שני רוצה למחוק את אותו קובץ שהמשתמש הראשון רוצה להוריד, מצב זה יכול להרוס את פעולת ההורדה של המשתמש הראשון ולפגוע במטרתו.

את בעיה זו נפתור לפי מנגנון מניעה הדדית, תחילה חשוב לציין שלכל לקוח שמתחבר לשרת נוצר לו Thread והוא פועל באופן עצמאי כך נוצר שישנם מספר משתמשים שיכולים להשתמש באותם משאבים שהם הקבצים, לכן נממש מבנה נתונים שבו נאחסן את שמו של הקובץ שבו יש שימוש כרגע, לפני שמכניסים אנו בודקים קודם שהקובץ לא נמצא כבר במבנה הנתונים (אם הוא נמצא ז"א שהוא בשימוש) ואנו לא מבצעים פעולה על הקובץ.

אם הואא נמצא אנו מכניסים ההכנסה מתבצעת ע"י Lock שלא יהיה מצב ששני משתמשים יהיו בתחרות להכנסת הקובץ ולאחר מכן נוכל לבצע את הפעולה שאנו רוצים על הקובץ.

ברגע שהמשתמש מסיים את הפעולה הוא מוציא את הקובץ ממבנה הנתונים. ישנם שני מבנה נתונים לשם כך אחד לקבצים ואחד לתיקיות.

5. ארכיטקטורת תוכנה:

התכונה מורכבת באמצעות צד שרת וצד לקוח.

התוכנה נכתבה בשפת Python גרסה 3.9.7 והסביבת פיתוח היא PyCharm. באחריות השרת לקבל חיבורים חדשים, לנהל את הקבצים שנמצאים אצלו והקבצים שהמשתמש מעוניין לעלות אליו. בצד הלקוח ניתן לבצע את פעולות על הקבצים/תיקיות כגון: שינוי שם, מחיקה, הזזה. ניתן לעלות קבצים/תיקיות וליצור תיקיות חדשת.

צד השרת:

הקמה של השרת בפועל, באמצעות sockets שהתקשורת מבוססת על פרוטוקול TCP, מכיוון שאנו רוצים להעביר קבצים על גבי רשת אמינה, ישנה פונקציה שתתואר בהמשך שפועלת במקביל כThread שמטרתה להמתין באופן פעיל לחיבור משתמשים לשרת.

בנוסף ישנה פונקציה בשרת שפועלת גם כן במקביל כThread שמטרתה ברגע שנוצר לכל לקוח Thread משלו בשרת, הפונקציה מחכה באופן פעיל לבקשות והודעות שנשלחות מצד הלקוח ומטפלת בהם בהתאם.

* כתובת השרת הברירת מחדל היא כתובת הIP של המחשב שעליו הוא רץ, והפורט שערכו 9999.
* במסך הפתיחה של הGUI של השרת יש אפשרות להזין את פרטי הIP והPort שאנו רוצים להקים את השרת או לבחור את העערכים הדיפולטיביים.
* השרת משתמש בתיקייה שנקראת server\_data שדרכה הוא מאחסן את הקבצים שמעלים אל השרת עושה שימוש גם בתיקייה בשם media שבה נמצא הICON של הGUI.
* לשרת יש בנוסף GUI שדרכו נוכל לראות איזה לקוח התחבר אליו, כלומר את כתובת הIP של ואת הפורט שלו ואיזה בקשה הוא שלח לשרת, פירוט הבקשת ומטרותיהן יפורט בהמשך.
* בGUI של השרת נוכל בנוסף לראות מספר המשתמשים הפעילים המחוברים לשרת וכתובת הIP והPort שלו.
* השרת והלקוח מתקשרים ביניהם באמצעות רשת TCP עם בקשות המוגדרות מראש מעין פרוטוקול ייחודי לגבי הפעולות שהמשתמש מבצע בתיקייה, על מנת לממש ולבצע את הפעולות: מחיקה, העלאה, הורדה, שינוי שם של קובץ/תיקייה,יצירת תת תיקייה חדשה ,תזוזה בין תת תיקיות לתיקיות האם, הצגת הקבצים הנמצאים באותו רגע בשרת, התחברות לשרת, והתנתקות מהשרת.
* בשרת ממומש מנגנון של מניעה הדדית, שמטרתו למנוע מקרה שבו למשל שמשתמש מוריד קובץ ובמהלך ההורדה, מחליט משתמש אחר למחוק את אותו הקובץ, על מנת שלא תופסק ההורדה ויתקבלו שגיאות בעקבות זאת, מובטח שתהליך ההורדה יהיה מוגן.עקרון המניעה ההדדית תקף לגבי הפעולות של הורדה, שינוי שם ומחיקה על מנת להבטיח תקינות הפעולות על קבצים במקביל ע"י מספר משתמשים שמחוברים לשרת, בכל מקרה של ניסיון הפרה תוצג למשתמש שמנה להפר הודעה בהתאם.
* בשרת בנוסף יש בדיקה של תקינות הקבצים שמתקבלים בשרת מהלקוח ע"י חישוב Checksum והשוואתם האם הם שווים. ברגע שלקוח רוצה לעלות קובץ לשרת הוא שולח את פרטי הקובץ והCS שלו, ברגע שהמידע מגיע לשרת הוא כותב את הקובץ בפועל אצלו ולאחר הכתיבה מבצע חישוב הCS שלו ומבצע השוואה עם הCS ששלח הלקוח, אם אכן שווים ההעלאה תבוצע בהצלחה, אם לאתוצג הודעת שגיאה בהתאם ללקוח.

צד הלקוח:

צד הלקוח הוא בעצם אפליקציית GUI שנכתבה באמצעות Tkinter, באפליקציה יש שני חלונות:מסך הפתיחה והמסך הראשי.

צד הלקוח עושה שימוש גם בתיקייה media שמשם הוא טוען את לוגו האפליקציה ותמונות נוספות.

במסך הפתיחה בדומה למסך הפתיחה אצל השרת, המשתמש מצג את פרטי הIP והPort של השרת אליו הוא רוצה להתחבר, יש גם אפשרות לבחור ערכים דיפולטיביים.

ברגע שנתחבר יוצג לנו באפליקציה במסך הראשי הנתיב של התיקייה, את פרטי הקבצים בדומה למבנה של תיקייה במחשב האישי וכפתורים לביצוע הפעולות השונות.

בנוסף תפועל פונקציה שעובדת במקביל לGUI כThread, שמטרתה לחכות באופן פעיל להודעה/בקשות מצד הלקוח וניהול שלהם שבהתאם להודעה/בקשות.

בחלק התחתן של החלון הראשי יוצגו לנו הכפתורים שבאמצעות נממש את הפעולות שנרצה על הקבצים ובתיקייה הכפתורים הם: open ,download, move to, upload folder ,upload file, delete, rename, ברגע שנלחץ עליהם תשלח בקשה לשרת מהלקוח.

* מטרת כפתור הRename – לשנות שם של קובץ/תיקייה לפי בחירתו של המשתמש.
* מטרת כפתור הDelete – למחוק קובץ/תיקייה לפי בחירתו של המשתמש.
* מטרת כפתור הUpload File – לעלות קובץ לפי בחירתו של המשתמש לשרת.
* מטרת כפתור הDownload – להוריד קובץ מהשרת לפי בחירתו של המשתמש.
* מטרת כפתור הOpen – לפתוח תת תיקייה לפי בחירתו של המשתמש.
* מטרת כפתור הUpload Folder – לעלות תיקייה לשרת לפי בחירות של המשתמש ממחשבו האישי.
* מטרת כפתור הmove to – להעביר/להזיז קבצים ותיקיות מהתיקייה שבה הם נמצאים אל תיקייה אחרת שבשרת.

בחלק העליון החלון הראשי של האפליקציה יוצג הנתיב העדכני שאנו נמצאים בו, וכפתורים עם תמונות של "חץ למעלה", כפתור עם תמונה של שני חצים בצורת מעגל שזה כפתור ה"רענון" וכפתור עם תמונה של "תיקייה", ברגע שנלחץ עליהם תשלח בקשה בהתאם לשרת.

* מטרת כפתור ה"חץ למעלה" – היא לנוע בין תתי תיקיות לתיקיות האם.
* מטרת כפתור ה"רענון" – היא להציג את הקבצים והתיקיות בפועל שנמצאים כרגע בשרת.
* מטרת כפתור ה"תיקייה" – היא ליצור תיקייה חדשה עם בחירת שם ע"פ בקשת המשתמש.
* ברגע שנעשה את אחת הפעולות תשלח הודעת בקשה לשרת, השרת יגיב בהתאם לבקשה, אם הבקשה תקינה היא תבוצע בשרת, אם לא תשלח הודעת שגיאה מהשרת ללקוח ובהתאם לסוג השגיאה תוצג הודעה ללקוח בהתאם.
* העברת הקבצים נעשת באמצעות chunks בגודל 1KB, הקבצים נקראים באופן בינארי ותוכנם המועבר הוא בינארי, הקובץ נוצר בהתאם לקצב העברת הchunks כלומר כל פעם 1KB עד שמגעים לגודל הקובץ.
* הבחירה בchunks נעשתה במטרה להקל על הרשת במקרה של העברת מידע גדול, באמצעות chunks נוכל לחלק את המידע לחתיכות קטנות ולהעבירן בכל פעם ובכך לא להעמיס על הרשת. מה שמאפשר לתוכנה לא להיות איטית או להיתקע.
* כמו כן נעשה שימוש בהשוואת Checksum על מנת להבטיח אם כתיבת הקובץ בשרת נעשתה נכון ותואמת לקובץ שהלקוח רצה לשלוח.
* כמו כן יש הגבלה של שם הקובץ/תיקייה ל30 תווים, גודל הפקטה היא 1024, ישנה בדיקה שאנחנו לא עוברים את גודל הפקטה כאשר יש מצב שבו אנחנו מעבירים נתיב ארוך.

נפרטי את סוגי הבקשות והההודעות שמועברות בין הלקוח לשרת:

* "UPLOAD" – בקשה להעלאת קובץ, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לעלות קובץ אליו, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בהעברת הקובץ אל השרת ו השרת מפעיל את הפונקציה שמיועדת לטפל בכתיבת הקובץ בפועל בשרת.
* "UPLOAD\_FOLDER" – בקשה להעלאת תיקייה ממחשבו האישי של המשתמש אל השרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לעלות תיקייה אליו, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בהעברת התיקייה אל השרת ו השרת מפעיל את הפונקציה שמיועדת לטפל ביצירת התיקייה ובכתיבת הקבצים שבתוכה בפועל בשרת.
* "MOVE" – בקשה להעברת קובץ/תיקייה מתיקיית המקור לתיקיית יעד אחרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה להעביר קובץ/תיקייה, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בהעברת הקובץ/תיקייה והשרת מפעיל את הפונקציה שמיועדת לטפל בהעברת הקובץ/תיקייה בפועל בשרת.
* "DOWNLOAD" – בקשה להורדת קובץ, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה להוריד קובץ ממנו, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בהעברת הקובץ מהשרת והשרת מפעיל את הפונקציה שמיועדת לטפל בשליחת הקובץ בפועל מהשרת.
* "LIST" – מטרתה של הבקשה להגיד לשרת שהלקוח מבקש לראות ולקבל את הקבצים והתיקיות שנמצאים אצלו כרגע, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בהעברת הקובץ מהשרת והשרת מפעיל את הפונקציה שמיועדת לטפל בשליחת הקובץ בפועל מהשרת. הבקשה בדרך כלל נשלחת באותו רגע התחברות לחלון הראשי לאפליקציה, נעשה בה שימוש ע"י בקשות "UPDATE" ו- "REFRESH".
* "RENAME" – בקשה שמטרתה לבקש שינוי שם של קובץ או תיקייה בשרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לשנות שם של קובץ או תיקייה, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בשינוי שם הקובץ או התיקייה בהשרת והשרת מפעיל את הפונקציה שמיועדת לטפל בשינוי שם הקובץ או התיקייה בפועל מהשרת.
* "DELETE" – בקשה שמטרתה לבקש מחיקה של קובץ או תיקייה בשרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לשנות למחוק קובץ או תיקייה, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו במחיקה של קובץ או תיקייה מהשרת והשרת מפעיל את הפונקציה שמיועדת לטפל במחיקה של קובץ או תיקייה בפועל מהשרת.
* "FOLDER" - בקשה שמטרתה לבקש ייצרה של תיקייה חדשה בשרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה ליצור תיקייה חדשה, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו ביצירה של תיקייה חדשה בשרת והשרת מפעיל את הפונקציה שמיועדת לטפל ביצירה של תיקייה חדשה בפועל מהשרת.
* "OPEN" – בקשה שמטרתה לבקש לפתוח תת תיקייה בשרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לפתוח תת תיקייה ולראות את הקבצים, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בפתיחת תת תיקייה בשרת והשרת מפעיל את הפונקציה שמיועדת לטפל פתיחת תת תיקייה בפועל מהשרת.
* "REFRESH" – בקשה זו דומה לבקשה LIST, אך היא נעשת באופן יזום ע"י המשתמש ע"י לחיצה על כפתור "הרענון", במטרה לראות לרענן ולראות את הקבצים והתיקיות בעדכניות בשרת, בגלל זה נבחר לה שם של בקשה חדשה ולא LIST. ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לפתוח לרענן, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בהצגת הקבצים והתיקיות בשרת והשרת מפעיל את הפונקציה שמיועדת לטפל בהצגת הקבצים והתיקיות בפועל מהשרת, הבקשה בנוסף עושה שימוש ב"LIST".
* "UPDATE" – מטרה של בקשה זו ברגע שבוצע עדכון/שינוי/יצירה/מחיקה בשרת, לאחר סיום אותן פעולות נשלחת בקשה זו מהשרת ללקוח, שמטרתה להגיד ללקוח שיש צורך בעדכון הנתונים שמוצגים לו כעת באפליקציה והלראות לו את השינוי שנעשה בפועל על ידו, הבקשה עושה שימוש בבקשת "LIST".
* "ARROW" – בקשה שמטרתה לבקש לעבור מתת תיקייה לתיקיית האם בשרת, ברגע שהשרת מקבל את הבקשה הוא מבין שהלקוח רוצה לפתוח לעבור מתת תיקייה לתיקיית האם, הוא מאשר ללקוח שקיבל בכך שהוא שולח את אותה בקשה ללקוח, בזמן הזה הלקוח מפעיל את הפונקציה שמיועדת לטפל אצלו בפתיחת תיקיית האם בשרת והשרת מפעיל את הפונקציה שמיועדת לטפל פתיחת תיקיית האם בפועל מהשרת.
* "DONE" – הודעה זו נשלחת במסגרת תהליך של טיפול של בקשות אחרות, מטרתה היא להעביר ללקוח שהטיפול הסתיים בהצלחה והתהליך הסתיים.
* "OK" – הודעה זו מטרתה להגיד לשרת שהלקוח סיים את תהליך טיפול הבקשה.
* "DISCONNECT" – בקשה שמטרתה להעביר בקשת ניתוק מהלקוח לשרת או מהשרת ללקוח, ברגע שהבקשה נשלחת מתבצע סיום החיבור בין השרת ללקוח ולהיפך, והחיבור הסתיים.

בד"כ הבקשה נשלחת באופן יזום ע"י לחיצה על כפתור X בחלון האפליקציה אצל השרת והלקוח.

נפרט את סוגי הודעות השגיאה:

* "IN\_USE" – מטרתה של ההודעה היא להגיד ללקוח שהקובץ/תיקייה נמצאים בשימוש ע"י משתמש אחר, הודעה זאת נשלחת מהשרת ללקוח.
* "NO\_MARKED" – מטרתה של ההודעה היא להגיד לשרת ששום קובץ/תיקייה לא סומן ואפשר לסגור את התהליך של הטיפול בבקשה.
* "NOT\_EXIST" - מטרתה של ההודעה להגיד ללקוח שהקובץ/תיקייה לא קיימים בשרת, הודעה זו נשלחת מהשרת ללקוח.
* "EXISTS" – הודעה זו נשלחת מהשרת ללקוח ומטרתה להגיד ללקוח שהקובץ/תיקייה שבהם הוא ורצה לעשות איזושהי פעולה, פתוחה ע"י משתמש אחר ולא ניתן לבצע שום פעולה.
* "ERROR" – מטרתה של ההודעה להגיד ללקוח שישנה שגיאה בתהליך טיפול הבקשה, למשל אם הCS לא שווה, או להיפך לשרת לדוגמה אם הלקוח לא בחר נתיב עבור הורדת הקובץ.
* "REPLACE" – הודעה זו מטרתה להגיד ללקוח שהקובץ שהוא רוצה לעלות כבר קיים בשרת והאם ברצונו להחליף אותו.

תרשים של הקמת החיבור וניתוק החיבור:

TCP Socket

Client

TCP Socket

Server

Bind

Listen

Connect

Close the connection

Close the connection

Disconnect request

Disconnect request

* בקו המקווקו שאר הפעילות של האפליקציה כגון בקשות להעלאה של קבצים, הורדה ועוד.

תרשים דוגמה של תקשורת בין השרת ללקוח להעלאת קובץ, כאשר המצב תקין (אין קובץ דומה שקיים במערכת ואין שגיאה בCS):

UPLOAD

LIST

UPDATE

OK

OK

Transfering the data file

OK

file name, file size, server path, file CS

UPLOAD

client

server

ברגע שמתחילה פעולת העלאה הלקוח והשרת מאשרים זה לזה שפעולת UPLOAD הולכת להתבצע, לאחר מכן המשתמש מתבקש לבחור קובץ, אם יסגור את חלון בחירת הקובץ ולא לחץ OK תשלח הודעת ERROR מהלקוח לשרת מכיוון שאין קובץ לבחירה והפעולה לא תבוצע. אם כן, הלקוח שולח לשרת את גודל הקובץ, שמו, הנתיב שבו אנו נמצאים כרגע אפליקציה והCS של הקובץ כעת לאחר שהפרטים הדגיעו לשרת יכול להיות מצב שיש קובץ דומה בשרת ואז תשלח הודעת REPLACE ללקוח אם ברצונו להחליף את הקובץ הקיים אם יאשר תתבצע העלאה ויוחלף הקובץ הקיים.

אם אין קובץ קיים אז ההעלאה תתרחש לאחר סיומה תתרחש בדיקת הCS של הקובץ שהועלה לשרת פועל אם תקין תשלח הודעת OK בין שני הצדדים ואז השרת שולח הודעת UPDATE על מנת להגיד ללקוח שהוא יכול לבצע בקשת LIST לצורך עדכון הצגת הקבצים אצלו, אם לא תשלח הודעת ERROR תוקפץ הודעת שגיאה בהתאם ללקוח בעקבות כך יופסק התהליך.

תרשים דוגמה לתהליך בקשת DOWNLOADתקין:

DOWNLOAD

DOWNLOAD

file name, path

DONE

Transfering the data file

OK

client

server

ברגע שמתחילה פעולת ההורדה הלקוח והשרת מאשרים זה לזה שפעולת DOWNLOAD הולכת להתבצע, תחילה תהיה בדיקה אלו קבצים סומנו ע"י המשתמש להורדה, אם בחר בתיקייה להורדה תוצג לו הודעת שגיאה. אם לא ימשיך התהליך לשלב הבא, ישלחו לשרת פרטי הקובץ על מנת להכניסו לקטע הקריטי ששאר המשתמשים ידעו שהקובץ כרגע בשימוש, ברגע שהשרת מקבל את פרטי הקובץ הוא בודק תחילה אם הקובץ הזה בשימוש כבר ע"י משתמש אחר אם כן שולח ללקוח הודעת IN\_USE ותוצג הודעת בצד להקוח למשתמש בהתאם.

אם לא תשלח הודעת OK שבו אפשר להמשיך בתהליך ותתבצע ההעברת של נתוני הקובץ להורדה, כמובן לאחר שבחר את המקום שבו הוא רוצה להוריד אותו לאחר שההורדה תסתיים תשלח הודעת DONE מהלקוח לשרת שתכריז על סיום התהליך.

ישנה בדיקה גם אם הקובץ המבוקש קודם קיים בשרת בכלל אם הוא לא קיים תשלח הודעה מהשרת ללקוח NOT\_EXIST, כמו כן ישנה בדיקה אם הלקוח סימן קובץ לפני, אם לא תשלח הודעה מהלקוח לשרת NO\_MARKED ויופסק התהליך.

תרשים דוגמה לתהליך בקשת LISTתקין:

LIST

path

LIST

client

server

Amount files

OK

Length's message per each file

OK

Transfering the data file

ברגע שמתחילה פעולת הצגת הקבצים הלקוח והשרת מאשרים זה לזה שפעולת LIST הולכת להתבצע, יוחלפו הנתונים להצגת הנתונים, הלקוח שולח תחילה את הנתיב שבו הוא נמצא על מנת להציג את הקצבים שנמצאים באותו נתיב בשרת לאחר מכן השרת ישלח ללקוח את כמות הקבצים לאחר מכן הלקוח מאשר קיבל, לאחר מכן השרת שולח את גודל המידע שהוא הולך להעביר לכל קובץ הלקוח מאשר שקיבל עם הודעת OK ולאחר מכן מתרחש העברת נתוני הקצבים והצגתם למשתמש.

תרשים דוגמה לתהליך בקשת RENAMEתקין:

RENAME

RENAME

file name, file type , path

client

server

OK

New name

OK

DONE

DONE

ברגע שמתחילה פעולת שינוי שם של קובץ הלקוח והשרת מאשרים זה לזה שפעולת RENAME הולכת להתבצע. תחילה ישלחו לשרת פרטי הקובץ שלו המשתמש רוצה לשנות את השם על מנת להכניסו לקטע הקריטי ששאר המשתמשים ידעו שהקובץ כרגע בשימוש, ברגע שהשרת מקבל את פרטי הקובץ הוא בודק תחילה אם הקובץ הזה בשימוש כבר ע"י משתמש אחר אם כן שולח ללקוח הודעת IN\_USE ותוצג הודעת בצד להקוח למשתמש בהתאם.

אם לא תשלח הודעת OK שבו אפשר להמשיך בתהליך, תוקפץ למשתמש חלון קטן שבוא יצטרך להכניס את השם החדש לקובץ, תהיה בדיקה אם המשתמש הכניס תווים חוקיים אם לא תוקפץ הודעת שגיאה בהתאם, אם הכל תקין ישלח השם החדש לשרת. השרת בתמורה יעדכן שקיבל באמצעות הודעתOK .

כעת יתבצע בפועל שינוי השם של הקובץ ולאחר השינוי שני הצדדים יאשור אחד לשני על סיום התהליך במאצעות הודעת DONE..

ישנה בדיקה גם אם הקובץ המבוקש קודם קיים בשרת בכלל אם הוא לא קיים תשלח הודעה מהשרת ללקוח NOT\_EXIST, כמו כן ישנה בדיקה אם הלקוח סימן קובץ לפני, אם לא תשלח הודעה מהלקוח לשרת NO\_MARKED ויופסק התהליך.

תרשים דוגמה לתהליך בקשת DELETEתקין:

DELETE

DELETE

File name, file's type, path

client

server

OK

DONE

DONE

LIST

ברגע שמתחילה פעולת מחיקת קובץ הלקוח והשרת מאשרים זה לזה שפעולת DELETE הולכת ללהתבצע. תחילה ישלחו לשרת פרטי הקובץ שאותו המשתמש רוצה למחוק, על מנת להכניסו לקטע הקריטי ששאר המשתמשים ידעו שהקובץ כרגע בשימוש, ברגע שהשרת מקבל את פרטי הקובץ הוא בודק תחילה אם הקובץ הזה בשימוש כבר ע"י משתמש אחר אם כן שולח ללקוח הודעת IN\_USE ותוצג הודעת בצד להקוח למשתמש בהתאם.

אם לא תשלח, ימשיך תהליך ותתבצע בפועל מחיקת הקובץ בשרת, לאחר שהשרת מחק את הקובץ תשלח הודעת OK ללקוח מהשרת, להקוח בתמורה יחזיר לשרת הודעת DONE שסיים את התהליך והשרת יאשר זאת ללקוח באמצעות שליחה נוספת מצידו, לאחר מכן תתחיל פעולת LIST מצד הלקוח על מנת לעדכן את תצוגת הקבצים.

ישנה בדיקה גם אם הקובץ המבוקש קודם קיים בשרת בכלל אם הוא לא קיים תשלח הודעה מהשרת ללקוח NOT\_EXIST, כמו כן ישנה בדיקה אם הלקוח סימן קובץ לפני, אם לא תשלח הודעה מהלקוח לשרת NO\_MARKED ויופסק התהליך.

תרשים דוגמה לתהליך בקשת OPENתקין:

OPEN

OPEN

path

client

server

DONE

LIST

ברגע שמתחילה פעולת פתיחת תיקייה הלקוח והשרת מאשרים זה לזה שפעולת OPEN הולכת ללהתבצע. תחילה ישלחו לשרת נתיב התיקייה, על מנת להכניסה לקטע הקריטי ששאר המשתמשים ידעו שתיקייה כרגע בשימוש, ברגע שהשרת מקבל את הנתיב הוא בודק תחילה אם הנתיב קיים אם לא קיים ישלח הודעת NOT\_EXIST ללקוח, אם כן קיים יבצע את הפתיחה של התקייה וישלח הודעת DONE לסיום התהליך ללקוח מנגד הלקוח יש לך בקשת LIST להצגת הקבצים של התיקייה שנפתחה.

תרשים דוגמה לתהליך בקשת FOLDERתקין:

FOLDER

FOLDER

Path, Folder's name

client

server

OK

DONE

DONE

LIST

ברגע שמתחילה פעולת יצירת תיקייה חדשה והשרת מאשרים זה לזה שפעולת FOLDER הולכת להתבצע. תחילה ישלחו לשרת פרטי התיקייה החדשה הנתיב שבו היא הולכת להיווצר והשם החדש שלה.

לאחר שהשרת קיבל את הפרטים הוא יבדוק אם תיקייה באותו נתיב עם שם דומה קיימת אם כן הוא ישלח הודעה ללקוח בשם EXISTS, אם לא יצור את התיקייה החדשה בפועל וישלח הודעת OK שהכל תקין.

כעת הלקוח ישלח הודעת DONE לסיום התהליך והשרת יחזיר לו בתמורה שקיבל ויסיים את התהליך גם אצלו.

ישנה בדיקה גם אם הקובץ הכניס שם חוקי כמו בRENAME. אם לא הכניס שם בכלל הפעולה תפסק ע"י שליחת הודעת מהלקוח לשרת בשם NO\_NAME.

תרשים דוגמה לתהליך בקשת ARROWתקין:

ARROW

ARROW

OK, path

client

server

DONE

UPDATE

LIST

ברגע שמתחילה פעולת לעלות למעלה בנתיב הלקוח והשרת מאשרים זה לזה שפעולת ARROW הולכת להתבצע. תחילה ישלחו לשרת נתיב התיקייה, השרת ברגע שמקבל את נתיב התיקייה הוא מוציא את התיקייה ממבנה הנתונים שמסמן שהיא בשימוש, תשלח הודעת DONE מהשרת שסיים את התהליך.

ולאחר מכן הודעת UPDATE שהלקוח יכול לשלוח בקשת LIST להצגת הקבצים של התקייה שאלי הוא עבר.

תרשים דוגמה לתהליך בקשת REFRESHתקין:

REFRESH

REFRESH

LIST

client

server

ברגע שמתחילה פעולת רענון הלקוח והשרת מאשרים זה לזה שפעולת REFRESH הולכת ללהתבצע.

לאחר מכן הלקוח שולח בקשת LIST לעדכון הצגת הקבצים אצלו לשרת.

תרשים דוגמה לתהליך בקשת MOVEתקין:

MOVE

MOVE

Path, file\_name, file\_path

client

server

OK/IN\_USE

Folder\_name

DONE

DONE

ברגע שמתחילה פעולת הזזת קובץ הלקוח והשרת מאשרים זה לזה שפעולת MOVE הולכת ללהתבצע. תחילה ישלחו לשרת פרטי הקובץ שאותו המשתמש רוצה להעביר, על מנת להכניסו לקטע הקריטי ששאר המשתמשים ידעו שהקובץ כרגע בשימוש, ברגע שהשרת מקבל את פרטי הקובץ הוא בודק תחילה אם הקובץ הזה בשימוש כבר ע"י משתמש אחר אם כן שולח ללקוח הודעת IN\_USE ותוצג הודעת בצד להקוח למשתמש בהתאם.

אם לא תשלח, ימשיך תהליך יוקפץ למשתמש חלון קטן שבו יצטרך להכניס את שם התיקייה שאליה הוא רוצה להעביר ותשלח שם התיקייה לשרת. כעת יתבצעו מספר בדיקות, האם המשתמש רוצה להעביר תיקייה לאותה התיקייה או שאם הוא רוצה להעביר קובץ/תיקייה לתיקייה שכבר הם נמצאים או שאם התיקייה החדשה קיימת בשרת תשלח הודעת NOT\_EXIST מהשרת על מנת להודיע כך ללקוח, תהיה בדיקה בנוסף אם הקובץ שאותו המשתמש רוצה להעביר לתיקייה החדשה אם יש קובץ כבר עם אותו שם, אם כן תשלח הודעת EXISTS מהשרת ללקוח.

כמו כן תהיה בדיקה אם המשתמש הכניס שם לחלון שבו הוא צריך לציין לאיזו תיקייה להעביר.

אם כל הבדיקות עברו בהצלחה אזי הלקוח והשרת יאשרו אחד לשני על סיום התהליך עם הודעת DONE.

כמו כן ישנה בדיקה אם הלקוח סימן קובץ לפני, אם לא תשלח הודעה מהלקוח לשרת NO\_MARKED ויופסק התהליך.

תרשים דוגמה לתהליך בקשת UPLOAD\_FOLDERתקין:

UPLOAD\_FOLDER

FOLDER

UPLOAD\_FOLDER

client

server

FOLDER

Path, folder name

OK/EXISTS

DONE/EXISTS

DONE/EXISTS

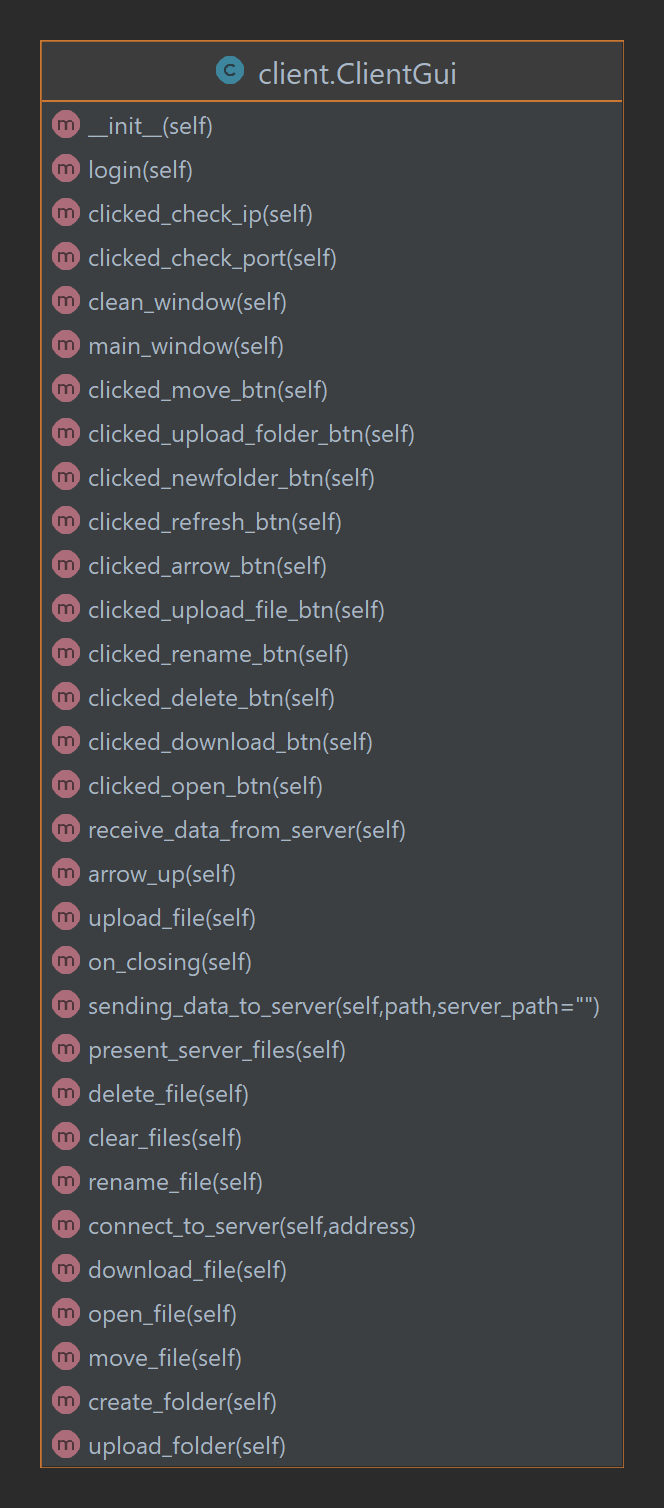
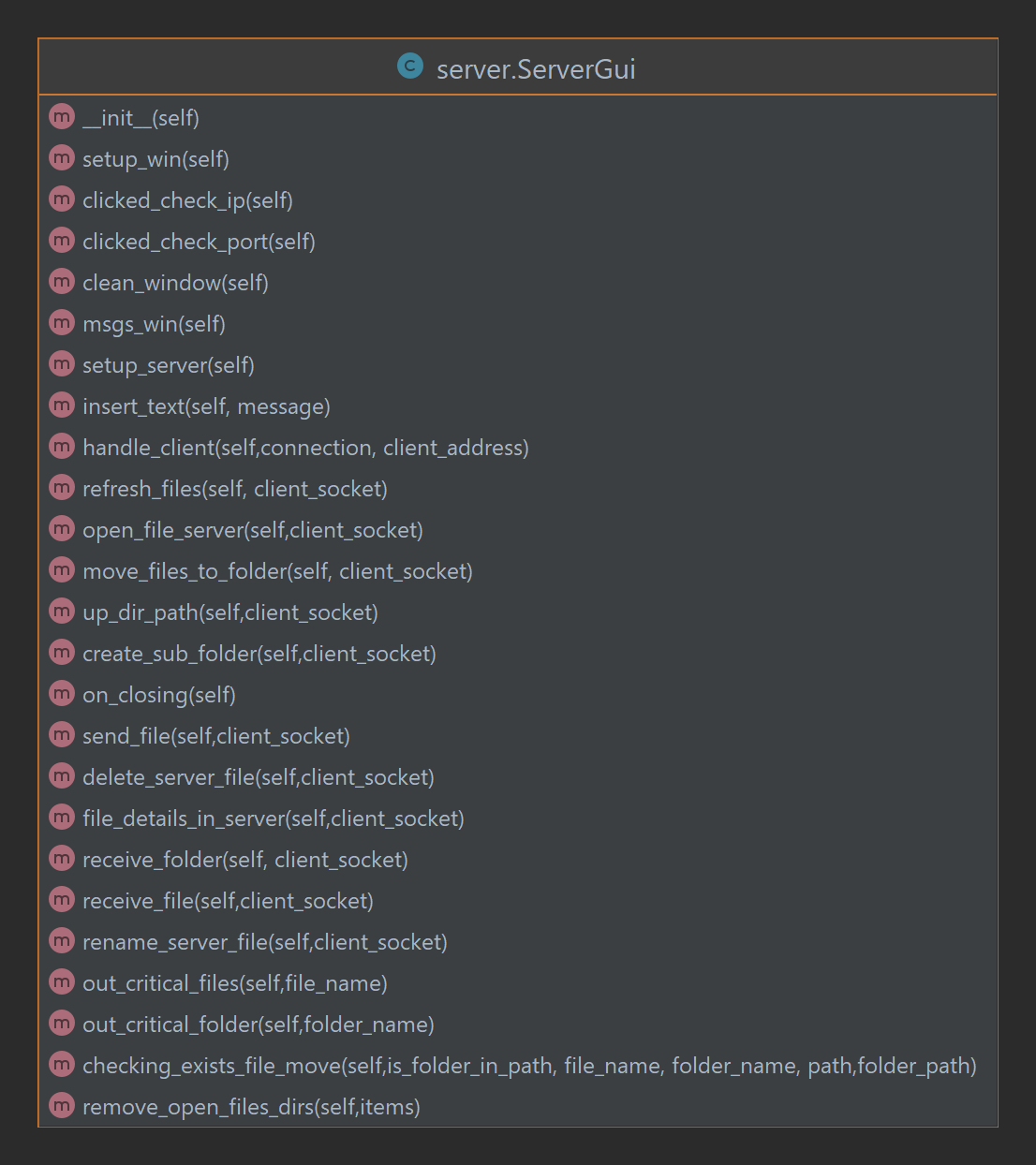
UPLOAD

UPLOAD

UPDATE

רגע שמתחילה פעולת העלאת תיקייה הלקוח והשרת מאשרים זה לזה שפעולת UPLOAD\_FOLDER הולכת להתבצע. לאחר מכן הלקוח משתמש בבקשת FOLDER על מנת ליצור תיקייה חדשה בשרת עם שם התיקייה שהמשתמש רוצה לעלות לשרת, הלקוח ישלח לשרת את הנתיב שבו התיקייה תעלה ואת שם התיקייה, בתמורה השרת יבדוק אם התיקייה כבר קיימת אם קיימת יודיע ללקוח באמצעות הודעת EXISTS אם לא ישלח הודעת OK ללקוח.

לאחר מכן תשלח הודעת DONE לאישור סיום תהליך יצירת התיקייה בין שני הצדדים ולאחר מכן הלקוח יעשה שימוש בבקשת UPLOAD להעלאת כל קובץ בתיקייה, לאחר סיום ההעלאה של כל קובץ תשלח הודעת UPDATE לצורך עדכון הצגת התיקייה החדשה שעלתה לשרת אצל הלקוח.

5.1 פירוט מחלקות ופונקציות:

|  |
| --- |
| auxiliary \_methods.py |
| + calculate\_size(bytes) |
| + check\_path(path) |
| + check\_file\_or\_folder(path) |
| + fix\_path(path) |
| + path\_up(file\_path) |
| + check\_illegal\_name(name) |
| + check\_length\_name(name) |
| + check\_move\_file\_folder(path, folder\_name) |
| + calculate\_checksum(file\_path) |

5.2 תיאור המחלקות:

|  |  |
| --- | --- |
| שם המחלקה | תפקיד |
| ClientGui | * מציגה את הGUI של האפליקציה (החלון והממשק). * כל ממשק ההתקשרות והלוגיקה מול השרת ממוש במחלקה. * כל הפונקציות של האפליקציה ממומשות במחלקה זו. |
| ServerGui | * הקמת השרת בפועל. * הקמת הGUI של השרת והצגת פרטיו בזמן אמת כגון בקשות, מס' משתמשים פעילים ועוד. * הארכיטקטורה של השרת טיפול בבקשות מהלקוח ותגובה אליהם בהתאם. * כתיבת הקבצים/תיקיות בפועל, שינוי שמם, מחיקתם ושליחתם. * ניהול כל הלוגיקה של השרת ומימוש בפועל של השרת. |

5.2 תיאור הפונקציות:

|  |
| --- |
| מחלקה: ClientGui |

|  |  |
| --- | --- |
| שם הפונקציה | תפקיד |
| \_\_init\_\_ | * אתחול ראשוני של משתני המחלקה. * אתחול של הגדרת חלון האפליקציה (רוחב,אורך , אייקון, שם) * הפעלת חלון האפליקציה. |
| login | * הגדרה של חלון הפתיחה של האפליקציה. * הצגת תמונת רקע, שדות טקסט לIP ולport, כפתורים לסימון, כפתור התחברות. |
| clicked\_check\_ip | * קליטת הקלט משדה הטקסט של הIP והבחנה האם המשתמש בחר בIP דיפולטיבי או לא. |
| clicked\_check\_port | * קליטת הקלט משדה הטקסט של הport והבחנה האם המשתמש בחר בport דיפולטיבי או לא. |
| clean\_window | * מנקה את כל הwidgets ממסך האפליקציה. |
| main\_window | * הצגת המסך הראשי של האפליקציה. * זה כולל את הלייבל של הנתיב שבו אנחנו נמצאים כרגע, הצגת פרטי הקבצים שנמצאים בשרת. * הכפתורים וטעינת האייקונים שלהם. * התחברות בפועל לשרת לאחר שהמשתמש לחץ על Login. * הפעלת Thread שימתין לקבלת הודעות מהשרת ולתקשורת בין השרת לאפליקציה (לקוח). |
| receive\_data\_from\_server | * מפועלת כThread שרץ במקביל לאפליקציה, שמטרתו להמתין באופן פעיל לקבלת הודעות מהשרת ולתקשורת בין השרת ללקוח. * הפונקציה מקבל את ההודעות מהשרת ופועלת בהתאם לבקשה. |
| clicked\_newfolder\_btn | * לאחר לחיצה על כפתור ה"תיקייה" (שמטרתו ליצור תיקייה חדשה) תשלח בקשה לשרת, שמטרתה ליצור את התיקייה בפועל בשרת. |
| clicked\_refresh\_btn | * לאחר לחיצה על כפתור ה"רענון" (שמטרתו להציג את הסטטוס העדכני של הקבצים והתיקיות בשרת) תשלח בקשה לשרת, שמטרתה ליצור את הצגת הקבצים והתיקיות שיש בפועל בשרת. |
| clicked\_arrow\_btn | * לאחר לחיצה על כפתור ה"חץ למעלה" (שמטרתו לעלות למעלה בנתיב לתיקיית האם) תשלח בקשה לשרת, שמטרתה ליצור את זאת בפועל. |
| clicked\_upload\_btn | * לאחר לחיצה על כפתור הUpload (שמטרתו לעלות קבצים לשרת) תשלח בקשה לשרת, שמטרתה להגיד לשרת שאנו מעוניינים לעלות קבצים לשרת. |
| clicked\_rename\_btn | * לאחר לחיצה על כפתור הRename (שמטרתו לשנות את שמם של הקבצים והתיקיות בשרת) תשלח בקשה לשרת, שמטרתה להגיד לשרת שאנו מעוניינים לשנות שם של קובץ או תיקייה. |
| clicked\_delete\_btn | * לאחר לחיצה על כפתור הDelete (שמטרתו למחוק תיקייה או קובץ) תשלח בקשה לשרת, שמטרתה להגיד לשרת שאנו מעוניינים למחוק קובץ או תיקייה. |
| clicked\_download\_btn | * לאחר לחיצה על כפתור הDownload (שמטרתו להוריד קובץ) תשלח בקשה לשרת, שמטרתה להגיד לשרת שאנו מעוניינים להוריד קובץ. |
| clicked\_open\_btn | * לאחר לחיצה על כפתור הOpen (שמטרתו להיכנס לתת תיקייה) תשלח בקשה לשרת, שמטרתה להגיד לשרת שאנו מעוניינים להיכנס לתת תיקייה והלציד את התיקיות והקבצים שנמצאים בה. |
| arrow\_up | * לאחר שנשלחה בקשת לעלות בנתיב למעלה לשרת, הפונקציה זו שולחת לשרת את פרטי הבקשה, כלומר את הנתיב המדוייק שאליו נעלה. * בין היתר אם תהיה שגיאה שתשלח מהשרת, תוצג לשרת ההודעה בהתאם. |
| upload\_file | * אחראי על הצגת החלון של בחירת הקובץ שאותו אנחנו רוצים לעלות לשרת וקורא לפונקציה שמטפלת בפועל בהעלאת הקובץ לשרת. |
| sending\_data\_to\_server | * קוראת את הקובץ שאותו בחרנו לעלות ושולחת לשרת. * בין היתר נשלח לשרת גם פרטי הקובץ, גודל, שמו, CS. * מקבלת הודעות מהשרת בהתאם לקובץ שאותו רצינו לעלות, אם הוא נמצא בשרת, אם הCS לא שווה, אם אנו רוצים להחליף את הקובץ שקיים בשם דומה במערכת ואם העלאה הצליחה. |
| on\_closing | * מטרתה לסגור את את האפליקציה, ברגע שהמשתמש לוחץ על X בחלון, באמצעות שליחת בקשת ניתוק לשרת וסגירה בפועל של האפליקציה ויציאה מהתוכנית. |
| present\_server\_files | * תפקידה להציג את נתוני הקבצים והתיקיות שיש בפועל בשרת למשתמש באפליקציה. |
| delete\_file | * בודקת איזה קובץ/תיקייה המשתמש סימן שברצונו למחוק ושולחת את פרטי הקובץ למחיקה לשרת. * בנוסף מטפלת בהתאם בהודעות שנשלחות כפידבק לבקשה והצגת הודעה בהתאם למשתמש, כגון: האם קובץ/תיקייה לא קיימים בשרת, האם קובץ/תיקייה נמצאים בשימוש ע"י משתמש אחר והאם המחיקה הצליחה. * בנוסף מציגה הודעת שגיאה האם המשתמש לחץ על כפתור הDelete ולא סימן קובץ. |
| clear\_files | * מטרתה למחוק את נתוני הקבצים והתיקיות שמוצגים למשתמש באפליקציה. |
| rename\_file | * מטרתה לבדוק לאיזה קובץ/תיקייה בחר המשתמש לשנות את השם. * הצגת חלון קטן שבו המשתמש מזין את השם החדש, נבדק גם האם השם מכיל תווים תקנים. * שליחת הפרטים של הקובץ/תיקייה בפועל לשרת. * טיפול בהודעות פידבק שנשלחות מהשרת בעקבות הבקשה והצגה למשתמש בהודעה, האם הקובץ קיים בשרת, האם הוא בשימוש ע"י משתמש אחר והאם השינוי בוצע בהצלחה. * בנוסף מציגה הודעת שגיאה האם המשתמש לחץ על כפתור הRename ולא סימן קובץ. |
| connect\_to\_server | * חיבור המשתמש בפועל לשרת, אם החיבור לא מתבצע בהצלחה תוצג הודעת שגיאה למשתמש. |
| download\_file | * מטרתה לבדוק איזה קובץ בחר המשתמש להוריד. * קריאת הפרטים של הקובץ שנשלחים מהשרת לשם כתיבת הקובץ בפועל אצל המשתמש במחשב. * טיפול בהודעות פידבק שנשלחות מהשרת בעקבות הבקשה והצגה למשתמש בהודעה, האם הקובץ קיים בשרת, האם הוא בשימוש ע"י משתמש אחר, האם הוא בחר להוריד קובץ והאם השינוי בוצע בהצלחה. * בנוסף מציגה הודעת שגיאה האם המשתמש לחץ על כפתור הDownload ולא סימן קובץ. |
| open\_file | * מטרתה לבדוק איזה תיקייה בחר המשתמש לפתוח. * שליחת פרטי התקייה והנתיב המעודכן של תת התיקייה לשרת. * טיפול בהודעות פידבק שנשלחות מהשרת בעקבות הבקשה והצגה למשתמש בהודעה, האם התיקייה קיימת בשרת, האם הוא בחר לפתוח קובץ והאם הבקשה בוצעה בהצלחה. * בנוסף מציגה הודעת שגיאה האם המשתמש לחץ על כפתור הOpen ולא סימן קובץ. |
| create\_folder | * מטרתה ליצור תיקייה חדשה בפועל בשרת, לפי בקשת המשתמש. * תוצג למשתמש חלון קטן שבו יוכל לרושם את השם החדש. * שליחת פרטי התיקייה החדשה, השם החדש והנתיב. * טיפול בהודעות פידבק שנשלחות מהשרת בעקבות הבקשה והצגה למשתמש בהודעה האם הבקשה בוצעה בהצלחה. |
| upload\_folder | * מטרת הפונקציה לטפל בבקשה המשתמש לעלות תיקייה ממחשבו האישי שמכילה קבצים אל השרת. * הפונקציה עושה שימוש בדומה לפונקציה create\_folder על מנת ליצור את התיקייה * הפונקציה עושה שימוש בפונקציה sending\_data\_to\_server שמטרתה לשלוח לשרת את הקבצים,שבתוך התיקייה. * טיפול בהודעות פידבק שנשלחות מהשרת בעקבות הבקשה והצגה למשתמש בהודעה האם הבקשה בוצעה בהצלחה. |
| move\_file | * מטרתה של הפוקנציה לטפל בבקשה של המשתמש להעברת קבצים/תיקיות לתיקייה אחרת בשרת. * מימוש מנגנון של מניעה הדדית על הקבצים/תיקיות שאנו רוצים להעביר. * טיפול בהודעות פידבק שנשלחות מהשרת בעקבות הבקשה והצגה למשתמש בהודעה האם הבקשה בוצעה בהצלחה. |

|  |
| --- |
| קובץ:auxiliary \_methods |

|  |  |
| --- | --- |
| שם הפונקציה | תפקיד |
| calculate\_size | * תפקידה לחשב את גודל הקובץ. * הפונקציה מקבל כפרמטר את גודל הבתים של הקובץ ומחשבת את הייצוג שלו לפי KB,MB,GB. |
| check\_path | * מקבלת כפרמטר נתיב ובודקת האם הוא קיים בפועל בשרת. |
| check\_file\_or\_folder | * מקבלת כפרמטר נתיב של תיקייה או קובץ ובודקת האם בנתיב שקיבלה האם זה קובץ או תקייה. |
| fix\_path | * תפקידה לתקן את הנתיב שמתקבל כפרמטר מאפליקציה לנתיב שהשרת יוכל לקבל. * Home תיקייה השורש שמוצגת באפליקציה שבפועל תיקייה השורש בשרת היא server\_data , זה התיקון שמתבצע בפונקציה ההתאמה של תיקיית השורש. |
| path\_up | * תפקידה לעדכן את הנתיב המעודכן לאחר שהמשתמש מבקש לעלות לתיקייה האם/שורש בנתיב. * הפונקציה מעדכנת את הנתיב ע"פ הנתיב שמקבל כפרמטר. |
| calculate\_checksum | * תפקידה לחשב את הקובץ את הchecksum של הקובץ שמקבלת כפרמטר. * החישוב מתבצע בעזרת הנתיב לקובץ. |
| check\_illegal\_name | * בדיקת תקינות תווים חוקיים בבחירת שם חדש לקובץ או תיקייה, כפי שווינדוס עובד. |
| check\_length\_name | * בדיקת אורך חוקי לשם תיקייה או קובץ. המקסימום 30. |
| check\_length\_path | * בדיקה האם הנתיב לא גדול יותר מגודל הפקטה שהיא 1024. |

|  |
| --- |
| מחלקה: ServerGui |

|  |  |
| --- | --- |
| שם הפונקציה | תפקיד |
| \_\_init\_\_ | * אתחול ראשוני של משתני המחלקה. * אתחול של הגדרת חלון האפליקציה (רוחב,אורך , אייקון, שם) * הפעלת חלון האפליקציה. |
| setup\_win | * הגדרה של חלון הפתיחה של האפליקציה. * הצגת תמונת רקע, שדות טקסט לIP ולport, כפתורים לסימון, כפתור התחברות. * הקמה בפועל של השרת אחרי הזנת הפרטים מהמשתמש. |
| clicked\_check\_ip | * קליטת הקלט משדה הטקסט של הIP והבחנה האם המשתמש בחר בIP דיפולטיבי או לא. |
| clicked\_check\_port | * קליטת הקלט משדה הטקסט של הport והבחנה האם המשתמש בחר בport דיפולטיבי או לא. |
| clean\_window | * מנקה את כל הwidgets ממסך האפליקציה. |
| msgs\_win | * הצגת המסך הראשי של הGUI של השרת. * זה כולל את פרטי השרת, הIP והPort. * מספר המשתמשים המחוברים לשרת בפועל. * הצגת ההיסטוריה של השרת. * הפעלת Thread שימתין לקבלת הודעות מהלקוח ולתקשורת בין השרת לאפליקציה (לקוח/משתמש). |
| setup\_server | * מפועלת כThread שרץ במקביל לאפליקציה, שמטרתו להמתין באופן פעיל לקבלת לחיבורים חדשים של משתמשים שרוצים להתחבר לשרת. * לכל חיבור חדש שנוצר עם משתמש נוצר Thread משלו בשל כך. |
| clicked\_insert\_text  \_btn | * הכנסת טקסט לחלון ההודעות של הGUI של השרת. |
| handle\_client | * זהו Thread שרץ במקביל, שמטרתו להמתין באופן פעיל להועות ולהתקשרות מצד הלקוח/משתמש. * ברגע שמתקבלת הודעה, השרת פועל בהתאם להודעה. |
| refresh\_files | * ברגע שנשלחת בקשה לרענון הקבצים, השרת מאשר שקיבל את הבקשה ושולח למשתמש להמשך התהליך. |
| open\_file\_server | * טיפול בפועל בבקשת Open שנשלחה מהמשתמש לשרת. * מקבל את פרטי הנתיב לפתיחת התיקייה מהלוקח ומחזיר פידבק האם הפעולה הצליחה או שהתיקייה לא קיימת. |
| up\_dir\_path | * טיפול בפועל בבקשת Arrow שנשלחה מהמשתמש לשרת. * מקבל את פרטי הנתיב לפתיחת תיקיית האם מהלקוח ומחזיר פידבק האם הפעולה הצליחה. |
| create\_sub\_folder | * טיפול בפועל בבקשת Folder שנשלחה מהמשתמש לשרת. * מקבל את פרטי הנתיב ושם התיקייה ליצירת התיקייה מהלקוח ומחזיר פידבק האם הפעולה הצליחה, או שהיא לא קיימת או שלא נבחר שם עבורה מצד הלקוח. |
| send\_file | * טיפול בפועל בבקשת Download שנשלחה מהמשתמש לשרת. * מקבלת את פרטי הקובץ מהלקוח לצורך הורדה של הקובץ מהשרת ומחזיר פידבק האם הפעולה הצליחה, או שהוא לא קיים או שלא נבחר שם עבורה מצד הלקוח או שנגרמה איזושהי שגיאה. |
| remove\_open\_files\_dirs | * מוציאה קבצים/תיקיות שהיו בשימוש מהקטע הקריטי. |
| delete\_server\_file | * טיפול בפועל בבקשת Delete שנשלחה מהמשתמש לשרת. * מקבלת את פרטי הקובץ/תיקייה מהלקוח לצורך מחיקתו של הקובץ/תיקייה מהשרת ומחזיר פידבק האם הפעולה הצליחה, או ש הקובץ/תיקייה לא קיימים או שהקובץ/תיקייה בשימוש ע"י משתמש אחר. |
| file\_details\_in\_server | * טיפול בפועל בבקשת List שנשלחה מהמשתמש לשרת. * השרת שולח בפועל למשתמש את פרטי הקבצים שנמצאים אצלו לצורך הצגתם של הקבצים ופרטיהם אצל המשתמש. |
| receive\_file | * טיפול בפועל בבקשת Upload שנשלחה מהמשתמש לשרת. * השרת מקבל מהלקוח את פרטי הקובץ שהוא רוצה לעלות לשרת, השרת מבצע את הכתיבה בפועל ולאחר שסיים לכתוב את הקובץ, מתבצעת בדיקת Checksum ונשלחת הודעה בהתאם ללקוח בעקבות תוצאות הבדיקה. |
| rename\_server\_file | * טיפול בפועל בבקשת Rename שנשלחה מהמשתמש לשרת. * השרת מקבל מהלקוח את פרטי הקובץ שהוא רוצה לשנות את שמו בשרת, השרת מבצע השינוי בפועל ומחזיר פידבק האם הפעולה הצליחה או שהקובץ כבר לא קיים בשרת או שהוא בשימוש ע"י משתמש אחר. |
| on\_closing | * מטרתה לסגור את השרת, ברגע שהמשתמש לוחץ על X בחלון, באמצעות שליחת בקשת ניתוק לשרת וסגירה בפועל של האפליקציה ויציאה מהתוכנית. |
| out\_critical\_files | * תפקידה להודיע על סיום הקטע הקריטי ושליפת הקובץ ממבנה הנתונים שמאחסן את הקבצים שבשימוש. |
| out\_critical\_folder | * תפקידה להודיע על סיום הקטע הקריטי ושליפת התיקייה ממבנה הנתונים שמאחסן את את שמות התיקיות שבשימוש. |
| receive\_folder | * תפקידה לטפל בבקשת העלאת תיקייה לשרת שנשלחת מהמשתמש. * מימוש מנגנון מניעה הדדית. * כתיבה בפועל ויצירת התיקייה בשרת והקבצים שבתוכה. * בדיקה האם תיקייה אם שם דומה קיימת, אם כן על המשתמש לבחור שם אחר. |
| move\_files\_to\_folder | * תפקידה לטפל בבקשה שנשלחה מהמשתמש להעברת תיקייה/קובץ לתיקייה אחרת בשרת. * מימוש מנגנון מניעה הדדית. * טיפול בשגיאות בזמן ההעברה, האם שם התיקייה שהמשתמש הזין קיימת, האם יש קובץ עם אותו שם בתיקיית היעד, האם המשתמש מעביר תיקייה לאותה תיקיית יעד ועוד. |

6. בדיקות וטסטים:

הבדיקות נעשו באופן ידני ולא בוצעו טסטים ידניים.

מטרתם של הבדיקות נועדו להבטיח את תקינות האפליקציה והשרת ויציבותם.

הבדיקות נעשו ע"י סימולציה של מספר משתמשים שמחוברים לשרת על מנת לבדוק מקרי קצה ומניפולציות מצד המשתמש.

הבדיקות נעשו על גבי הרשת המקומית, על ווינדוס 10 ו-11, בנוסף הקמת שרת במחשב אחד ולקוח במחשב אחר שמתחבר אל השרת, כלומר בדיקת התקשורת ותקינות האפליקציה בשני מחשבים שונים.

הבדיקות שנעשו כדי להבטיח את תקינות פעולת האפליקציה ויציבותה:

שלב ההתחברות:

1. בשלב ההתחברות כאשר המשתמש לא מזין פרטי IP וPort על מנת להתחבר לשרת, שהתוכנית לא תקרוס.
2. האם הפונקציה של בחירת IP כברירת מחדל והPort כברירל מחדל אכן עובד ויש התחברות לשרת, תקף גם לשרת האם אכן הוקם השרת בהצלחה.
3. האם הזנת פרטי הIP והPort מצד המשתמש כרצונו, אכן עובד ויש התחברות לשרת, תקף גם לשרת האם אכן הוקם השרת בהצלחה.
4. אם לא הצליחה ההתחברות מכל סיבה תוצג הודעת שגיאה בהתאם למשתמש ויהיה לו נסיונות נוספים להתחבר ללא קריסה של האפליקציה, תקף גם לשרת.

לאחר ההתחברות:

1. בדיקה שאכן במסך הראשי של האפליקציה לאחר ההתחברות מוצגים הפרטים הנכונים כגון: הנתיב ופרטי הקבצים:גודלם, שמםו תאריך שינוי אחרון שנמצאים בשרת.
2. תקינות פעולת הבקשת ברגע שלוחצים על הכפתור שאכן הפעולה עובדת למשל: יצירת תיקייה חדשה, לעלות בנתיב למעלה לתיקיית האם, רענון של הקבצים בשרת והצגתם ללקוח, העלאה של קובץ לשרת ע"י המשתמש, הורדה של קובץ מהשרת לבחירת המשתמש, מחיקת קובץ מהשרת ע"י בחירה של המשתמש, שינוי שם של קובץ ע"י בחירה מהמשתמש ופתיחת תת תיקייה והצגת הקבצים/התיקיות שבתוכה.
3. בדיקת מקרי קצה של הבקשה להעלאה של קבצים לשרת:
   * 1. כשהמשתמש לא בוחר קובץ להעלאה.
     2. בדיקה שהקובץ שנבחר ע"י המשתמש אכן זה אותו קובץ שנכתב בשרת הבדיקה היא באמצעות Checksum.
     3. בדיקה בפועל שהקובץ שהתקבל בשרת אכן עובד ופועל.
4. בדיקת מקרי קצה של הבקשה להורדה של קבצים מהשרת:
5. כשהמשתמש לא בוחר קובץ להורדה.
6. כשהמשתמש לא מסמן קובץ להורדה ולוחץ על כפתור הDownload.
7. כשמשתמש אחר מנסה למחוק את הקובץ או לשנות את שמו של הקובץ בזמן שמשתמש אחר מוריד את הקובץ.
8. בדיקה בפועל שהקובץ אכן קיים בשרת.
9. בדיקה שאכן הקובץ שהורד זה אכן הקובץ והוא תקין ופועל.
10. בדיקת מקרי קצה של הבקשה לשינוי שם של קובץ בשרת:
11. כשהמשתמש לא בוחר קובץ מסמן קובץ ולוחץ על כפתור הRename.
12. כאשר המשתמש לא בוחר שם חדש לקובץ.
13. האם המשתמש הזין שם חוקי לקובץ.
14. בדיקה האם הקובץ נמצא בשימוש ע"י משתמש אחר.
15. בדיקה האם הקובץ קיים בפועל בשרת.
16. בדיקה שאכן השינוי בוצע בהצלחה בשרת וגם מוצג למשתמש באפליקציה.
17. בדיקת מקרי קצה של הבקשה למחיקת קובץ בשרת:
18. כשהמשתמש לא בוחר קובץ/תיקייה ולוחץ על כפתור הDelete.
19. בדיקה האם הקובץ/תיקייה נמצאים בשימוש ע"י משתמש אחר.
20. בדיקה האם הקובץ/תיקייה קיימים בפועל בשרת.
21. בדיקה שאכן השינוי בוצע בהצלחה בשרת וגם מוצג למשתמש באפליקציה.
22. בדיקת מקרי קצה של הבקשה לפתיחת תת תיקייה בשרת:
23. כשהמשתמש לא בוחר תיקייה ומבקש לפתוח קובץ.
24. האם המשתמש לא סימן תיקייה ולחץ על כפתור הOpen.
25. בדיקה האם התיקייה אכן קיימת בפועל בשרת.
26. בדיקה האם בפועל מתרחשת פתיחת התיקייה והקבצים/תיקיות שבתוכה נכונים ומוצגים למשתמש באפליקצה.
27. האם לאחר פתיחת תת התיקייה עודכן הנתיב הנכון.
28. בדיקת מקרי קצה של הבקשה ליצירת תיקייה חדשה בשרת:
29. כאשר המשתמש לא בוחר שם לתיקייה.
30. בדיקת תקינות תווים חוקיים בבחירת השם לתיקייה בחדשה שתווצר.
31. בדיקה האם התיקייה קיימת ונוצרה בפועל בשרת.
32. בדיקה שאכן השינוי מוצג למשתמש באפליקציה.
33. בדיקת מקרי קצה של הבקשה של הבקשה להעלאה בנתיב בשרת:
34. כאשר המשתמש נמצא בתיקיית השורש ולוחץ על כפתור ה"חץ למעלה".
35. כאשר אנחנו המשתמש נמצא בתיקיית שורש כלשהי האם הבקשה מבוצעת בפועל ומתעדכן בנוסף הנתיב נכון.
36. בדיקה האם תיקיית האם קיימת בשרת.
37. בדיקה שאכן השינוי מוצג למשתמש באפליקציה.
38. בדיקת מקרי קצה של הבקשה לרענון של הקבצים בשרת:
39. בדיקה שאכן הקבצים העדכניים בשרת מוצגים למשתמש באפליקציה.
40. בדיקה של תקינות פרטי הקבצים המוצגים למשתמש באפליקציה.
41. בדיקת מקרי קצה של הבקשה להעברת קבצים של הקבצים בשרת:
42. בדיקה שהתיקייה שאליה המשתמש רוצה להעביר אכן קיימת.
43. בדיקה שאם בתיקיית היעד יש קבצים אם אותו שם לקבצים שאנו רוצים להעביר תקף גם לגבי תיקיות.
44. אם המשתמש לוחץ על הכפתור אך לא בוחר בשום קובץ/תיקייה.
45. אם המשתמש לא הכניס שם לתיקיית היעד.
46. אם משתמש אחר מנסה לעשות פעולה על הקבצים שנבחרו בזמן העברתם.
47. בדיקה שהמשתמש מבקש להעביר את הקבצים מתיקיית המקור שלהם לתיקיית המקור לשהם.
48. בדיקה אם המשתמש מבקש להעביר תיקייה ומגדיר את תיקיית היעד באותו שם אם התיקייה שהוא רוצה להעביר.
49. בדיקת מקרי קצה של הבקשה להעלאת תיקייה לשרת:
50. בדיקה שאכן התיקייה נוצרה בפועל והקבצים שבתוכה נוצרו ותקינים.
51. בדיקה האם ישנה תיקייה בשם דומה לתיקייה שאנו מעלים.
52. כאשר המשתמש לא בוחר תיקייה.
53. בדיקה שאכן בשרת מוצגים כמות המשתמשים המחוברים בפועל, פרטיהם של פירוט הבקשות שנשלחות מצד הלקוח שמחובר לשרת, מי הלקוח שהתחבר לשרת ופרטיו של השרת הIP והPort.

סיום ההתחברות:

1. בדיקה שכאשר המשתמש לוחץ על הכפתור X בחלון האפליקציה, שאכן האפליקציה נסגרת ומתבצע בקשת ניתוק מהשרת והחיבור נותק בהצלחה מצד השרת.
2. בדיקה שאכן בשרת מתבצע ניתוק הלקוח כלומר מוצג לנו שהלקוח עם פרטיו התנתק מהשרת וכמות המשתמשים המחוברים לשרת מתעדכנת בהתאם.
3. בדיקה שכאשר המשתמש לוחץ על הכפתור X בחלון הGUI של השרת אכן השרת נסגר ונשלח הודעת ניתוק ללקוחות שמחוברים לשרת והניתוק מהם בוצע בהצלחה ללא שגיאות וקריסות.

7. פערים במצב הקיים:

1. האפליקציה אינה תומכת בקבצים עם שמות בעברית או שחלק משמם בעברית, בעקבות אי תמיכה של שפת Python.

2. אם הנתיב שמוצג בחלון הראשי של האפליקציה גודל מאוד, יתכן ולא יראה סוף הנתיב, מכיוון שהצגת הנתיב ממומש בLabel.

3. אין בדיקה האם בשרת יש מקום מספיק פנוי לכתיבת קבצים חדשים.

4. אין תמיכה של העלאת מספר קבצים במקביל, בכל פעם רק קובץ אחד, אם רוצים לעלות כמה במקביל אפשר לשים אותם תחת תיקייה ולעלות את התיקייה.

5. אין אפשרות למיין את הקבצים לפי שמות,גודל או תאריך שינוי אחרון.

6. אין אפשרות לעלות תיקייה שיש בה תיקייה אלא רק קבצים, על מנת להתגבר על זה, אפשר לדחוס את לZIP והתוכנה מתייחסת לזה כקובץ וניתן לעלות לשרת.

7. ניהול הרשאות גישה.