

אפיון פרויקט

הגנת סייבר

**מגיש: עומר כפיר (330869017) תיכון הרצוג כפר סבא יב'3**

**שם פרויקט: Silent Net**

**שם מנחה: אופיר שביט**

**שם חלופה: הגנת סייבר ומערכות הפעלה**

**תאריך הגשה: 00/05/2025**

תוכן עניינים

[**מבוא** 4](#_Toc197020911)

[**תיאור כללי** 4](#_Toc197020912)

[**מטרות ויעדים** 4](#_Toc197020913)

[**בעיות תועלות וחסכונות** 5](#_Toc197020914)

[**סקר שוק – חקירת פתרונות קיימים** 5](#_Toc197020915)

[**סקירת טכנולוגיית הפרויקט** 7](#_Toc197020916)

[**תיחום פרויקט** 7](#_Toc197020917)

[**פירוט תיאור המערכת (אפיון)** 8](#_Toc197020918)

[**תיאור מפורט של המערכת** 8](#_Toc197020919)

[**פירוט על יכולות שהיא תעניק לכל סוג משתמש** 9](#_Toc197020920)

[**פירוט יכולות המערכת** 10](#_Toc197020921)

[**בדיקות, לו"ז וסיכונים** 17](#_Toc197020922)

[**פירוט בדיקות שמתוכננות לפרויקט** 17](#_Toc197020923)

[**תכנון וניהול לוז** 20](#_Toc197020924)

[**סיכונים** 21](#_Toc197020925)

[**תיאור ארכיטקטורה המערכת** 23](#_Toc197020926)

[**תיאור החומרה** 23](#_Toc197020927)

[**מראה גרפי של קשרי החומרה** 24](#_Toc197020928)

[**תיאור טכנולוגיות רלוונטיות** 25](#_Toc197020929)

[**שפות תכנות** 25](#_Toc197020930)

[**מערכת הפעלה** 26](#_Toc197020931)

[**תחומי עניין** 29](#_Toc197020932)

[**תיאור זרימת המידע במערכת** 30](#_Toc197020933)

[**האזנה שקטה על מחשב המשתמש** 30](#_Toc197020934)

[**איסוף מידע בזמן אמת מן מחשב המשתמש** 31](#_Toc197020935)

[**גיבוי נתונים אצל המשתמש בעת שרת כבוי** 32](#_Toc197020936)

[**שליחת מידע המשתמש אל המנהל** 33](#_Toc197020937)

[**איסוף המידע במחשב השרת ל DB** 34](#_Toc197020938)

[**הצגת המידע באופן נגיש במחשב המנהל** 35](#_Toc197020939)

[**הגדרת רמת בטיחות אצל המנהל** 36](#_Toc197020940)

[**ניסוח וניתוח של הבעיה האלגוריתמית** 37](#_Toc197020941)

[**אלגוריתמים קיימים לפתרון הבעיה** 37](#_Toc197020942)

[**סקירת הפיתרון הנבחר** 37](#_Toc197020943)

[**הפנייה למקור רלוונטי** 37](#_Toc197020944)

[**תיאור סביבת הפיתוח** 38](#_Toc197020945)

[**תיאור פרוטוקול התקשורת** 39](#_Toc197020946)

[**דוגמה להודעות:** 39](#_Toc197020947)

[**טבלת הודעות פרוטוקול** 40](#_Toc197020948)

[**תיאור מסכי המערכת** 42](#_Toc197020949)

[**מסך פתיחה** 42](#_Toc197020950)

[**מסך הגדרות** 43](#_Toc197020951)

[**מסך ראשי** 44](#_Toc197020952)

[**מסך אישי** 44](#_Toc197020953)

[**מסך יציאה** 45](#_Toc197020954)

[**מסך טעינה** 46](#_Toc197020955)

[**מסך שגיאה** 46](#_Toc197020956)

[**מסך 404** 47](#_Toc197020957)

[**דיאגרמת מסכים** 48](#_Toc197020958)

[**תיאור מבני נתונים** 49](#_Toc197020959)

[**סקירת חולשות ואיומים** 51](#_Toc197020960)

[**מימוש הפרויקט** 55](#_Toc197020961)

[**מודולים/מחלקות מיובאים** 55](#_Toc197020962)

[**מודולים/מחלקות מקוריים** 57](#_Toc197020963)

[**manager.py** 57](#_Toc197020964)

[**server.py** 60](#_Toc197020965)

# **מבוא**

**ייזום**

## **תיאור כללי**

הפרויקט הינו פלטפורמה שמאפשרת למנהל של חברה להשגיח על פעילות לקוחותיו מרחוק ולוודא שהם מבצעים את העבודה כראוי. הפלטפורמה על המחשבים של העובדים מסתירה את עצמה כך שהעובדים בחברה לא יוכלו להסיר את התוכנית מהמחשב שלהם לבד או לשבש פעילותה. הפרויקט מחולק לשלושה חלקים – העובדים, המנהל והשרת שמנהל את התקשורת כנגד שני הרכיבים האחרים. הפרויקט יעזור למנהל לנתר את פעולות עובדיו ולראות אילו עובדים פועלים בפועל ואילו לא.

בחרתי בפרויקט זה מכיוון שהוא אינו פשוט ויהווה לי אתגר ואף בנוסף הוא עוסק בעיקרו בעולם מערכות ההפעלה, עולם שמרתק אותי. הפרויקט דורש למידה על התנהגות של מערכת ההפעלה והתעסקות עם קוד של מערכת ההפעלה. המטלה הינה לא פשוטה ומביאה עימה מספר אתגרים: הסתרת הליך במערכת הפעלה והאזנה לפעולות שונות מבלי פגע ויכולות של המשתמש לדעת, העברת מידע אמין למחשב המנהל אשר משמש בתור שרת, תצוגה גרפית בצורה נוחה אצל מחשב המנהל ועוד...

## **מטרות ויעדים**

לפרויקט מספר מטרות, חלק מן המטרות עוסקות בכל הנוגע למערכות הפעלה והכוונה להסתיר באופן מוחלט מן העובד בחברה את אותו תהליך בכדי שלא יוכל להסיר אותו/לשנות אותו, בו זמנית המנהל בניגוד מוחלט יוכל לקבל מידע מדויק ממנו אשר יופיע לו בצורה נוחה וקלה להתממשקות. על המערכת להיות נגישה כמה שיותר למנהל החברה וביחד עם זאת להיות מוסתרת כמה שיותר מהעובד בחברה שהתוכנה רצה אצלו במחשב.

## **בעיות תועלות וחסכונות**

המערכת באה לחסוך זמנים בשביל מנהלים בחברות. חברות אשר ישתמשו במערכת זו יוכלו באופן נגיש להאזין למחשבים של העובדים בחברה שלהם מבלי שהעובדים יוכלו לעשות דבר כנגד לכך. העובדים לא יוכלו למצוא או לשנות את התוכנית המאזינה בכדי "לעבוד" על המנהל שלהם, המנהל יידע על שלל פעילות המחשב שלהם מבלי יוצא מן הכלל (כמובן הפרטים החשובים, לא האזנה מלאה לכל event). המערכת תפתור את הבעיה שנוצרת אצל עובדים בחברות שאינם מנצלים את זמן העבודה באופן מיטבי ואף גם לעיתים לא מבצעים את העבודה הצפויה מהם במהלך העבודה. זוהי הינה בעיה חמורה שמפסידה כסף רב לחברות, וכאן בדיוק מגיעה המערכת שלי.

מבלי ידיעת העובדים המנהל יכול בכל רגע נתון להסתכל על נתוני המחשב של העובד אצלו בתוכנה, ומכך להבין אם העובד מבצע את עבודתו כראוי או שצריך להעיר לו ואף אם הדבר מתמשך לפטר את העובד מהחברה. המערכת תנגיש את המידע על הלקוחות השונים למנהל של החברה בכדי שיוכל לתצפת על עובדיו באופן נוח ויעיל, מבלי שיצטרך להתקשות עם תוכנה מסובכת, כמו שידוע כיום הרבה תוכנות בעלות פונקציונליות רבה מתקשות בלשמור על ממשק משתמש נוח וקל לשימוש, ולכן המערכת תדע להציג את המידע באופן נוח מבלי שמנהל החברה יצטרך להתקשות איתה.

## **סקר שוק – חקירת פתרונות קיימים**

כידוע כיום עולם אבטחת המידע הוא נושא מבין הנושאים שמעסיקים אותנו ביום יום. קיימים מאות אלפי אם לא מיליוני כלים באינטרנט לאבטחת סייבר. כלים אשר מדמים את פעולות של המערכת הם כלים כמו Anti-Virus/Anti-cheat. אותם כלים לרוב יהיו KLM (Kernel Loadable Module) והם יודעים להאזין לפעילויות המערכת, קלטים שונים, תהליכים כאלו ואחרים, ואף להתריע ולשלוח לאדם חיצוני בעת זיהוי סכנה.

דרך פעולה זו מדמה את פעולות המערכת בפרויקט שלי, אשר המערכת יודעת לשבת בליבת מערכת ההפעלה, ולשלוח מידע לאדם חיצוני, האומנם המערכת שלי אינה מנתחת את פעילות המחשב ומתריעה רק בעת סכנה, אך היא יודעת להאזין לליבת מערכת ההפעלה ולשלוח את המידע לאדם חיצוני. Silent Net מתמקדת לא רק בהאזנה דיסקרטית אלא גם במניעת חשיפה שעלולה לאפשר התערבות חיצונית.

המערכת מתוכננת כך שהמשתמש לא יוכל להשיג כל מידע מהותי שיאפשר לו לזהות, לאתר או לחסום את פעילותה. לשם כך, Silent Net שמה דגש על טשטוש סימנים מזהים כמו שמות מודולים, נתיבי קבצים, כתובות תקשורת ופרטים נוספים. בכך, גם אם המשתמש ינסה לנתח את המערכת, לא יימצאו רמזים ברורים שיאפשרו לו להפעיל אמצעים נגדיים. גישה זו מייחדת את Silent Net ממערכות אבטחה קלאסיות, שכן המטרה אינה להתריע על סכנות אלא לשמר פעילות סמויה לאורך זמן מבלי לאפשר גילוי או שיבוש.



## **סקירת טכנולוגיית הפרויקט**

במערכות הפעלה קיימת שונות רבה ומשמעותית, הן בין מערכות הפעלה שונות לחלוטין והן בין גרסאות שונות של אותה מערכת הפעלה. שונות זו באה לידי ביטוי בממשקי המערכת, במנגנוני ליבה, בתמיכה בחומרה, ובתצורת ניהול המשאבים.

כתוצאה מכך, קיימת חשיבות רבה לסביבה בה תורץ מערכת כלשהי, שכן ייתכן שתכונות מסוימות שהמערכת נשענת עליהן אינן זמינות או אינן מתפקדות באותו אופן בכל גרסה או מערכת. במקרה של המערכת הנוכחית, שמבצעת פעולות קריטיות בלב מערכת ההפעלה, יש קושי ממשי להפוך אותה לדינמית וגמישה כך שתוכל לרוץ על טווח רחב של גרסאות או מערכות.

משום כך, קיימת תלות חזקה בסביבת ההרצה, והמערכת מחויבת לפעול על גרסה מסוימת ומדויקת של מערכת ההפעלה שנבחרה מראש. התאמה זו נדרשת אך ורק בצד הלקוח – יש להדגיש כי צד השרת אינו כפוף לאותם מגבלות, והוא חופשי לפעול על מערכות אחרות. כדי לאפשר הרצה עקבית ונכונה של המערכת, ניתן להשתמש בטכנולוגיות כגון מכונות וירטואליות (VM), שיאפשרו יצירת סביבה זהה בכל הרצה.

כך ניתן להבטיח את יציבות המערכת ופעולתה התקינה, תוך בידוד מגורמים חיצוניים שיכולים להשתנות בין מחשב למחשב. לפיכך, הפרויקט כולו מחויב לרוץ על אותה גרסה בה פותח, ואין אפשרות או תמיכה בהרצה על גרסאות אחרות, אפילו אם הן שייכות לאותה מערכת הפעלה. כתוצאה מכך, כל מחשב שעליו תרוץ המערכת חייב להיות מותאם מראש לגרסת מערכת ההפעלה הספציפית שנבחרה במהלך תהליך הפיתוח.

## **תיחום פרויקט**

הפרויקט עוסק במגוון רחב של תחומים טכנולוגיים, שכל אחד מהם מהווה רכיב מרכזי בתפקוד ובמבנה של המערכת. להלן פירוט התחומים המרכזיים בהם יעסוק הפרויקט:

* + מערכות הפעלה: הפרויקט מעמיק בעבודה מול מערכת ההפעלה, תוך שימוש בטכניקות מתקדמות הכוללות ביצוע hooking לפונקציות קריטיות במערכת. טכניקות אלו יאפשרו יירוט ושינוי של התנהגות הפונקציות על פי צורכי המערכת. בנוסף, תתבצע הסתרה מכוונת של מידע מפני תהליכים אחרים, במטרה לשמור על חשאיות הפעולה של המערכת. ביצוע פעולת hooking מאפשר בנוסף קבלת מידע מאותם פונקציות קריטיות ושליחתו אל השרת.
  + רשתות תקשורת: כחלק מתהליך העברת המידע מהצד של המשתמש אל צד השרת ובתקשורת רציפה בין המנהל והשרת, המערכת תשתמש בפרוטוקולי רשת, ובפרט בפרוטוקול TCP. בחירה בפרוטוקול זה נובעת מהצורך בהעברת מידע אמינה ומהימנה, תוך הבטחת שלמות הנתונים בעת ההעברה. המידע שייאסף ממערכת ההפעלה ישודר בצורה סדירה ומבוקרת אל מחשב השרת, אשר יקלוט, יאחסן ויעבד את הנתונים בהתאם לדרישות המערכת.
  + הצפנה ואבטחת מידע: לצורך שמירה על סודיות ואבטחת המידע בזמן ההעברה, המערכת תשתמש בשיטות הצפנה שונות. הצפנה זו נועדה למנוע חשיפת המידע על ידי גורם שלישי שינסה ליירט את התעבורה בין מחשב הלקוח לשרת. השימוש בהצפנה יבטיח כי המידע הרגיש הנאסף לא ייחשף לגורמים בלתי מורשים, וכי התקשורת בין רכיבי המערכת תתבצע בצורה מאובטחת.
  + נושאים בהם הפרויקט לא עוסק: יש להדגיש כי המערכת אינה עוסקת בתחום של אבטחת מערכות מחשוב או בהגנה מפני תוכנות זדוניות, אנטי-וירוסים, פיירוולים וכדומה. כמו כן, המערכת לא נועדה לשם ניהול סיסמאות או שמירתן במסדי נתונים (DB). כלומר, אין מדובר בפתרון אבטחה כולל, אלא במערכת ניטור ומעקב פנימית, הפועלת באופן דיסקרטי למטרות ניהול ובקרה מצד גורמים בעלי הרשאה.

# **פירוט תיאור המערכת (אפיון)**

## **תיאור מפורט של המערכת**

המערכת המדוברת היא מערכת האזנה שקטה ובלתי נראית, אשר פועלת על מחשבי קצה המריצים את מערכת ההפעלה לינוקס. מטרת העל של המערכת היא לפקח על פעולתם התקינה והשוטפת של המחשבים, תוך שמירה על סודיות מוחלטת וללא כל אפשרות למשתמשי המערכת לזהות את נוכחותה. המערכת תופעל ברקע ותנטר בצורה שקטה מתודות שונות המבוצעות על ידי תוכניות שונות שרצות על המחשב, וזאת במטרה לאפשר מעקב רציף, מדויק ויסודי אחרי כל פעולה המתבצעת במערכת.

באמצעות מנגנוני האזנה מתקדמים, המערכת תוכל לעקוב אחרי מגוון רחב של פעולות, ביניהן הרצת תוכניות חדשות, יצירת ושימוש בחיבורים לרשת, ועוד. המידע שייאסף יאפשר לקבל תמונה מלאה, מקיפה ורציפה על כל מה שמתרחש במחשב בזמן אמת, כך שכל פעילות תדווח למחשב השרת והמנהל יוכל לבחון זאת בעצמו ולהחליט על עבודת העובד.

הייחודיות של המערכת טמונה בכך שהיא לא רק מבצעת מעקב אלא גם שומרת על עצמה מוסתרת לחלוטין. גם אם עובד יידע מראש על קיומה, הוא לא יוכל לזהות אותה, להסיר אותה או להפריע לפעולתה. לשם כך, המערכת תשתמש בטכניקות חכמות ומתקדמות, כמו שינוי מודולים קיימים של מערכת ההפעלה, שינוי דינמי של קריאות מערכת והתחמקות מגילוי או אמצעי ניטור אחרים.

המערכת תעבוד באופן קבוע לאיסוף מידע, אשר ייארז בצורה מסודרת, מאורגנת ויעילה, תוך שמירה על הפרדה בין משתמשים שונים. המידע יישלח למחשב שרת מרכזי שאיתו המנהלים יוכלו ליצור תקשורת ולהשיג את הנתונים ממנו השמורים על כל מחשב ומחשב, הנתונים שמורים בצורה ברורה, נוחה ונגישה. המידע יוצג לפי משתמשים ומחשבים, עם אפשרויות סינון מתקדמות, הצגת נתונים בזמן אמת, כולל הפקת דו"חות מפורטים לפי הצורך.

באופן כללי, המערכת מהווה פתרון טכנולוגי מתקדם שמפשט ומייעל את ניהול העובדים והמשאבים הדיגיטליים של הארגון. בזכות אוטומציה של תהליכי הפיקוח ושמירה על דיסקרטיות מוחלטת, המערכת מאפשרת למנהל לקבל שליטה מלאה, לצפות בהתנהלות העובדים, לאתר בעיות במהירות, וכל זאת תוך שמירה על רמת ביצועים גבוהה מאוד וללא השפעה ניכרת על המערכת הנבדקת.

## **פירוט על יכולות שהיא תעניק לכל סוג משתמש**

היכולות העיקריות שמספק הפרויקט ממוקדות בצד המנהל ולא בצד המשתמש (העובד). כפי שהוסבר קודם, מדובר בפרויקט שמטרתו לאסוף מידע על פעילות מחשבים של עובדים בארגון, תוך הסתרת פעילותו ככל האפשר. כלומר, אחד מיעדי הליבה של הפרויקט הוא לגרום לכך שהתוכנה תרוץ ברקע מבלי שהעובדים יהיו מודעים לקיומה. גם במקרה שהעובד כן מבחין בסימנים כלשהם המעידים על קיום התוכנה – לא תהיה לו היכולת להשפיע על פעולתה או להסירה.

לכן, מבחינת העובד, אין כל ממשק או תועלת ישירה מן המערכת, והיא אינה מספקת עבורו כל פונקציונליות. לעומת זאת, מנהל העובדים הוא הגורם שמפיק את מרב התועלת מהפרויקט. למנהל תהיה גישה למגוון רחב של יכולות, אשר יאפשרו לו לצפות בפעילות מחשבי העובדים בכל רגע נתון. הוא יוכל לבדוק האם עובדיו פועלים בהתאם להוראות העבודה, לזהות דפוסי התנהגות חריגים או בלתי יעילים, ואף להסיק מסקנות לגבי איכות הביצוע של כל עובד.

הגישה לנתונים מתקבלת בצורה נוחה וישירה על מחשב המנהל, כשהמידע נשלח ומעודכן בזמן אמת, מה שמאפשר תגובה מיידית או תכנון שיפורים בתהליכי העבודה. יכולת זו תורמת תרומה משמעותית לייעול ניהול העובדים ולשיפור אפקטיביות העבודה בארגון.

בנוסף, מכיוון שהעובדים אינם מודעים למעקב, הם אינם יכולים לדעת מתי ובאיזה אופן מתבצעת הבקרה על פעילותם. מצב זה מעניק למנהל יתרון משמעותי: הוא מקבל תמונה אותנטית של ההתנהלות בפועל, ללא ניסיון של העובדים להסתיר או לזייף ביצועים. באמצעות הנתונים הללו, המנהל יכול לבצע הערכה אובייקטיבית של תרומת העובדים לארגון, לקבל החלטות מבוססות מידע לגבי שיפור נהלים, ארגון מחדש של משימות ואף קבלת החלטות קשות כגון פיטורים – במקרים בהם מתגלה כי עובדים מסוימים אינם ממלאים את תפקידם כנדרש לאורך זמן.

להלן פירוט מפורט יותר של היכולות.

## **פירוט יכולות המערכת**

1.האזנה שקטה על מחשב המשתמש.

2.איסוף מידע בזמן אמת מן מחשב המשתמש.

3.גיבוי נתונים אצל המשתמש בעת שרת כבוי.

4.שליחת מידע של מחשב המשתמש אל מחשב השרת.

5.איסוף המידע במחשב השרת אל תוך DB.

6.הצגת המידע באופן נגיש במחשב המנהל.

7.הגדרת רמת בטיחות אצל המנהל.

|  |  |
| --- | --- |
| האזנה שקטה על מחשב המשתמש | שם היכולת |
| האזנה שקטה על מחשב המשתמש מאפשרת לכך שאין באפשרות העובדים למחוק/לשבש דברים בתוכנה בזמן פעילותה, כלומר התוכנה תמיד תפעל ואך ורק המנהל הוא בעל היכולת לשנות דברים בתוכנה. בכך המנהל יכול בכל זמן נתון לתצפת על עובדיו מבלי שיוכלו "לעבוד" על התוכנית ולגרום לה לשלוח מידע לא אמין. | מהות היכולת |
| **אצל הלקוח**   * ביצוע hookים שונים על פעולות מבלי לפגוע במהות הפעולות * הסתרת התהליך * האזנה "קלה" כך שאינה משפיע על ביצועי המערכת של מחשב המשתמש | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל * מחשב לקוח (עם מערכת הפעלה לינוקס) * הרשאות של המנהל על מחשב הלקוח | אובייקטים נחוצים |

|  |  |
| --- | --- |
| איסוף מידע בזמן אמת מן מחשב המשתמש | שם היכולת |
| איסוף מידע בזמן אמת מן המחשב של המשתמש מאפשרת ליכולות אחרות בפרויקט זה, אסיפת המידע היא יכולת קריטית אצל מחשבי המשתמשים, והיא חלק גדול מפרויקט זה. איסוף המידע בזמן אמת יאפשר למנהל לתצפת על עובדיו בזמן אמת (ולא רק לקבל סיכום), כלומר בכל רגע נתון המנהל בעל יכולת לבדוק את עובדיו ואת עבודותיהם. | מהות היכולת |
| **אצל הלקוח**   * ביצוע hookים שונים על פעולות מבלי לפגוע במהות הפעולות * שמירת המידע בצורת טקסט שניתן לקרוא ולהבין בצורה מונגשת * יצירת סדר במידע, כל פעולה שונה במחשב המשתמש עליה להיות מסודרת לפי סוג הפעולה. | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל * מחשב לקוח * קבצים/באפרים לאחסון המידע באופן זמני | אובייקטים נחוצים |

|  |  |
| --- | --- |
| גיבוי נתונים אצל המשתמש בעת שרת כבוי | שם היכולת |
| בזכות היכולת של הלקוח לאסוף בזמן אמת נתונים, עליו גם לאחסן אותם אצלו כאשר קיימות תקלות כאלו ואחרות בתקשורת או אפילו מחשב השרת נכבה. יכולת זה מכפה על תקלות כאלו שאינן בשליטת הפרויקט ובאות לפצות עליהן. גיבוי הנתונים אינו אינסופי אך מוגבל בגודלו על ידי המנהל בכדי לא להעמיס על מחשבי הלקוחות, הגודל נקבע לפי רצון המנהל. אם מחשב הלקוח נכבה אף הוא בעצמו מבלי לסגור את התוכנית באופן שלם, התוכנית תדע בעצמה לחזור למצב המקורי של גיבוי המידע, כלומר אין איבוד מידע גם כאשר מחשב הלקוח בעצמו נכבה בבורטליות. | מהות היכולת |
| **אצל הלקוח**   * ביצוע hookים שונים על פעולות מבלי לפגוע במהות הפעולות * שמירת המידע בצורת טקסט שניתן לקרוא ולהבין בצורה מונגשת * יצירת סדר במידע, כל פעולה שונה במחשב המשתמש עליה להיות מסודרת לפי סוג הפעולה. * התממשקות עם קבצים – פתיחה, כתיבה, קריאה, סגירה (ותוך כדי להישאר במצב Kernel). | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * KLM – Kernel Loadable Module, כלומר תהליך בהרשאות הגבוהות ביותר במחשב – קרנל * מחשב לקוח * קבצים לאחסון המידע באופן זמני | אובייקטים נחוצים |

|  |  |
| --- | --- |
| שליחת מידע של מחשב המשתמש אל מחשב השרת | שם היכולת |
| שליחת מידע של מחשב המשתמש אל מחשב השרת מאפשרת לשרת/מנהל לקבל את המידע על עובדיו ובכך מקשר את מחשבי העובדים למחשב השרת. המידע שנאסף על המחשבים נשלח בזמן אמת אל השרת ובכך התוכנית יכולה לבצע יכולות רבות על המידע שמתקבל מן המשתמשים. | מהות היכולת |
| **אצל הלקוח**   * תוכנית שמאזינה לפעולות הלקוח. * שליחת המידע מן מחשב המשתמש לשרת.   **אצל השרת**   * איסוף המידע. * קבלת המידע אצל מחשב השרת ופענוח המידע. | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * מחשב לקוח * Socket * מחשב שרת * קבצים/באפרים לאחסון המידע באופן זמני | אובייקטים נחוצים |

|  |  |
| --- | --- |
| איסוף המידע במחשב השרת אל תוך DB | שם היכולת |
| איסוף המידע במחשב השרת אל תוך DB מאפשר למחשב השרת לשמור את המידע הנשלח מן הלקוחות בזמן אמת ולהפריד אותו לכל לקוח בנפרד. כלומר כל מחשב של עובד בחברה יהיה שמור בתוך ה DB של כלל העובדים, כך שיהיה ניתן להפריד בין העובדים השונים במערכת. | מהות היכולת |
| **אצל השרת**   * איסוף המידע מן הלקוח. * פענוח המידע לפי הפרוטוקול * הכנסה ל DB את הערכים של הלקוח ממנו נשלחה ההודעה. | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * מחשב לקוח * Socket * מחשב שרת * טבלת DB אצל השרת | אובייקטים נחוצים |

|  |  |
| --- | --- |
| הצגת המידע באופן נגיש במחשב המנהל | שם היכולת |
| הצגת המידע באופן נגיש במחשב המנהל מאפשרת למנהל של העובדים באופן יעיל להסתכל על המידע של מחשבי העובדים ובכך לקבוע את יעילות עבודתם ואם הם עובדים כראוי או שאינם מבצעים את עבודתם כלל, ולכן חשוב להציג את המידע באופן מסודר כך שהמנהל יוכל להסתכל בנוחות ובזריזות על המידע. | מהות היכולת |
| **אצל המנהל**   * איסוף המידע מן השרת. * פענוח המידע לפי ההצפנה המוסכמת על ידי השרת והמנהל. * פענוח המידע לפי הפרוטוקול. * הצגה באופן ויזואלי על ידי GUI את נתוני ה DB בזמן אמת. | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * מחשב לקוח * Socket * מחשב שרת * מחשב מנהל * טבלת DB אצל השרת * GUI במחשב המנהל * מפתחות הצפנה | אובייקטים נחוצים |

|  |  |
| --- | --- |
| הגדרת רמת בטיחות אצל המנהל | שם היכולת |
| כאשר המנהל מתחבר לשרת לאחר ביצע תהליך Login עליו להגדיר את רמת הבטיחות שהינו רוצה בעת הפעלת המערכת אצלו, הכוונה היא שביכולתו להגביל את כמות הלקוחות שמחוברים בו זמנית לשרת ואף לתת לשרת רמת בטיחות שהשרת יפעל על ידו – רמת בטיחות תהיה כמות ההודעות שהשרת מסכים שאינן עובדות לפי הפרוטוקול שאיתו השרת מתקשר לפני שהשרת מנתק את לקוח זה. | מהות היכולת |
| **אצל המנהל**   * הצגת המידע באופן ויזואלי – GUI * התחברות ותקשורת עם השרת * הצפנת הודעות מקצה לקצה עם השרת   **אצל השרת**   * הצפנה הודעות מקצה לקצה עם המנהל * הגבלת כמות לקוחות * הגבלת כמות הודעות "בלתי חוקיות" לפי רמת בטיחות | אוסף פעולות/יכולות הנדרשות ליכולת הנ"ל |
| * Socket * מחשב שרת * מחשב מנהל * GUI במחשב המנהל * מפתחות הצפנה * אובייקטים של לקוחות בכדי שהשרת יוכל לספור לכל לקוח כמה הודעות "בלתי חוקיות" נשלחו ממנו | אובייקטים נחוצים |

# **בדיקות, לו"ז וסיכונים**

## **פירוט בדיקות שמתוכננות לפרויקט**

בפרויקט הזה אבצע מספר בדיקות כדי לוודא שהמערכת מתפקדת כפי שתוכננה. הבדיקות יתמקדו בהעברת נתונים וביכולת המעקב אחר פעולות המשתמש, תוך שמירה על יציבות המערכת. ביצוע הבדיקות הינו שלב הכרחי בפיתוח הפרויקט שבלעדיו לא ניתן להכריע אם הפרויקט עובד כראוי

**1.בדיקת העברת נתונים לשרת המרכזי**

|  |  |
| --- | --- |
| מטרה | לבדוק שהנתונים מהמחשב של העובד מועברים לשרת של המנהל באופן אמין |
| איך אבדוק | אגדיר שרת מדומה ואעקוב אחרי זרימת הנתונים ממחשבי העובדים. אשתמש בכלים לבדיקת חבילות רשת כדי לוודא שהמידע מגיע בשלמותו ובתזמון הנכון. |

**2. בדיקת עקביות הנתונים**

|  |  |
| --- | --- |
| מטרה | לוודא שהנתונים הנאספים ממחשבי העובדים משקפים את הפעולות שבוצעו במדויק. |
| איך אבדוק | אבצע פעולות מוגדרות מראש במחשב העובד, כמו פתיחת תוכנות ועבודה על מסמכים, ואשווה את הנתונים שנשלחו לשרת לאירועים בפועל. כך אוכל לראות שהמערכת עוקבת בצורה מדויקת אחר הפעולות. |

**3.בדיקת השפעה על ביצועי המערכת**

|  |  |
| --- | --- |
| מטרה | לבדוק שהמערכת לא פוגעת בביצועים של מחשבי העובדים |
| איך אבדוק | אבצע מדידת ביצועים (CPU, זיכרון) לפני התקנת המערכת ואחריה, ואשווה את התוצאות. כך אוכל לוודא שהמערכת לא מכבידה על המשאבים של המחשב ומאפשרת עבודה חלקה. |

**4. בדיקת אבטחת הנתונים**

|  |  |
| --- | --- |
| מטרה | לוודא שהתקשורת בין המנהל לשרת מוגנת. |
| איך אבדוק | אשתמש בהצפנה להעברת המידע ואבחן פרוטוקולי תקשורת מאובטחים (AES DH) על מנת לוודא שהנתונים נשארים חסויים. ומוגנים בפני האזנות של מחשבים ברשת. |

**5. בדיקת הגבלת מספר לקוחות ורמת בטיחות**

|  |  |
| --- | --- |
| מטרה | לוודא שביכולתו של המנהל להגביל כרצונו את כמות הלקוחות ולקבוע להם רמת בטיחות. |
| איך אבדוק | אחבר מנהל אל השרת ואתן מספר מאוד נמוך של לקוחות, אבדוק שכאשר מעל המספר המוגדר שהמנהל שלח של לקוחות מנסים להתחבר השרת לא יתן להם להתחבר. בנוסף אצור לקוח פיקטיבי שישלח מידע לא לפי הפרוטוקול ואבדוק אם השרת מנתק את החיבור עם לקוח |

**6. בדיקת גיבוי המידע אצל הלקוח**

|  |  |
| --- | --- |
| מטרה | לוודא שהלקוח מגבה מידע באופן אמין וביכולתו לגבה מידע בזמן אמת באופן מוגבל על ידי המנהל. |
| איך אבדוק | בזמן פעילות הלקוח אכבה את השרת ואמחוק את הנתונים שהיו שומרים עד כה אצל השרת. כאשר אדליק מחדש את השרת אבדוק אם יתקבלו נתונים שתואמים את פעילות הלקוח בזמן שהשרת היה כבוי. |

**7. בדיקת החבאתו של תוכנת הלקוח**

|  |  |
| --- | --- |
| מטרה | לוודא שהתוכנה המופעלת אצל הלקוח הינה מוסתרת מן הלקוח ואין ביכולתו לשבש את פעילותה |
| איך אבדוק | אפעיל את התוכנה אצל הלקוח ואבדוק על ידי תוכנות ופקודות שונות אם התוכנה נראית ללקוח – Wireshark, netstat, lsmod. החבאה מתוכנות ופקודות אלו ימנע מן הלקוח לשבש את פעילות התוכנה (לדוגמה - ללמוד איך התוכנית עובדת לפי הפקטות היוצאות ולחכות את התנהגות התוכנה ובכך לשבש את התוכנית). |

## **תכנון וניהול לוז**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **מספר** | **פעילות** | **תאריך יעד** | **תאריך בפועל** | **הערות** |
| **1** | הצעת רעיון פרויקט גמר | 17.10.2024 | 14.10.2024 | הצעת הפרויקט הינה פרויקט זה, לא התבצעו שינויים ברעיון הפרויקט המקורי |
| **2** | הצעה לפרויקט גמר | 30.10.2024 | 24.10.2024 | תכנון ראשוני של הפרויקט, הכלים והדרך ביצוע |
| **3** | מסמך אפיון – הגשת ביניים ראשונה | 10.11.2024 | 6.11.2024 | הגשה חלקית של אפיון הפרויקט |
| **4** | מסמך אפיון – הגשת ביניים שנייה | 20.11.2024 | 14.11.2024 | הגשה חלקית של אפיון הפרויקט |
| **5** | מסמך אפיון – הגשה סופית | 30.11.2024 | 22.11.2024 | הגשה סופית של אפיון הפרויקט המלא |
| **6** | הוכחת יכולת | 15.01.2025 | 12.01.2025 | ביצוע מעקב אחר פעולות מרכזיות ושליחה לשרת |
| **7** | ממשק גרפי | 31.01.2025 | 30.01.2025 | ממשק גרפי מלא אצל המנהל |
| **8** | פעילות מלאה של המנהל | 01.03.2025 | 27.02.2025 | המנהל יכול לעקוב אחרי העובדים ולנתר את עבודתם |
| **9** | פעילות מלאה של הלקוח | 20.03.2025 | 22.03.2025 | פיתוח מלא של קוד הלקוח כולל כל הפיצ'רים |
| **10** | סיום כתיבת הפרויקט | 31.03.2025 | 28.3.2025 | פיתוח מלא של הפרויקט ללא באגים |
| **11** | סיום בדיקות הפרויקט | 17.04.2025 | 9.04.2025 | בדיקות מלאות של הפרויקט כמו שצוין באפיון |
| **12** | תיק פרויקט | 30.04.2025 | דרוש לשנות |  |
| **13** | הגשת פרויקט גמר | --NULL-- | דרוש לשנות |  |

## **סיכונים**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **הסיכון** | **תיאור הסיכון** | **רמת סיכון** | **תיאור דרכי התמודדות** | **מה בוצע בפועל** |
| אי עמידה בלוח הזמנים | הגשה מאוחרת וחוסר עקביות בלוח זמנים המתוכנן. אי עמידה בלוח הזמנים משפיע על כל תהליך הפרויקט | קל | לדאוג לעקוב אחר הלוח זמנים. | דאגתי לעקוב אחר הלוח זמנים. |
| המידע שמועבר אינו מוצפן | המידע שמועבר מן המנהל לשרת אינו מוצפן ולכן אנשים שאינם חלק מהמערכת יכולים לפענח את ההודעות ולהשתמש בהם לטובתם | קשה | להשתמש בהצפנות שונות, הצפנה אסימפטרית להעברת המפתחות בשביל הצפנה סימטרית בשביל העברת ההודעות | השתמשתי בהצפנות בין המנהל לשרת. הסכמה על המפתחות מתבצעת בתחילת התקשורת בין השניים. |
| ממשק משתמש לא יציב ולא נוח לשימוש | ממשק המשתמש GUI אינו נוח לשימוש מה שמקשה רבות על השימוש בפרויקט מצד השרת. | בינוני | שינוי ממשק המשתמש כך שגם מי שאינו מבין את תוכן הפרויקט יוכל להבין איך להשתמש בממשק המשתמש, בדיקה כנגד אנשים חיצוניים | יצרתי ממשק משתמש שנוח לשימוש ואף נראה נוח לעין, אינו מסובך ויוצר הגיון בקרב משתמשים אפילו כשאינם מבינים בעולם המחשבים. |
| קריסת השרת | השרת נפל בשל סיבה לא ידועה כזו או אחרת | קשה | שימוש באמצעי זהירות בכדי למנוע את קריסתו של השרת (try/except), מזעור של קריסת השרת כך כאשר באמת יקרוס השרת כבר זו תהיינה בעיה חיצונית שהפרויקט לא אחראי עליה | קוד השרת ממזער את קריסתו על ידי כתיבתו כך שאם נתקל בבעיה כלשהי כזו או אחרת בזכות מנגנון try/except יתפוס את התקלה וידפיסה למסך. בנוסף הקוד הראשי של השרת מופרד מההתנהלות נגד כל לקוח בזכות threads, כלומר אם thread שמטפל בלקוח קורס, השרת ימשיך לפעול ולקבל לקוחות אחרים. |
| עומס יתר על השרת | השרת מוצף בלקוחות ומנהל מול מספר רב של לקוחות session מה שמכביד עליו ויקשה על מחשב השרת לתפקד כראוי | קשה | הגבלתם של מספר הלקוחות שיכולים להתחבר לשרת, מספר קבוע מקסימלי שיקבע מספר גג של לקוחות שיכולים בו זמנית לנהל session מול השרת | כאשר השרת מופעל, המפעיל את השרת יגדיר כמות דיפולטיבית של מקסימום לקוחות שיכולים להתחבר, לאחר מכן בעת התחברות המנהל יוכל אף הוא בעצמו להגביל את כמות הלקוחות (ההגבלה גם היא בטווח, בין 1-40 לקוחות). |
| DB של הלקוחות גדול מדי | ה DB שהשרת שומר בזמן אמת על לקוחותיו יגדל בצורה כה משמעותית מה שיכביד על השרת וביצועיו | קשה | מחיקת מידע אצל כל משתנה כאשר כמות המידע השמורה אצלו בטבלה עברה מכסה מסוימת שנקבעה מראש | במקום לשמור כל הודעה המתקבלת על ידי הלקוח כתבתי את הקוד כך שימספר כל הודעה, מה שמצד אחד חוסך כמויות גדולות של אחסון, ומצד שני חוסך גם כוח מחשוב כאשר מנסים לשלוף מידע מן DB. |
| הלקוח משנה את המערכת אצלו במחשב | הלקוח הצליח לשנות את המערכת ששולחת מידע מהמחשב אל השרת, מה שיפגע בתפקוד הפרויקט | קשה | הסתרת התוכנית כך שתעבוד מאחורי הקלעים, עבודה עם ה kernel בכדי להחביא את התוכנית. בנוסף תדאג להסתרת כל ביצועיה, שליחת המידע למחשב חיצוני גם כן הוא יוסתר על ידי התהליך. | הסתרתי את התוכנית בפני המשתמש, אין ביכולת המשתמש לבצע פעולות כנגד התוכנית משום שאין לו שום מידע עליה (הלקוח לא יכול לדעת על איזה פורט השרת מתקשר וסוג ההודעות הנשלחות אל השרת). |
| תקשורת לא אמינה | הלקוח והשרת מתקשרים באופן לא אמין, לא כל המידע שהלקוח שולח מגיע בשלמותו אל השרת | קשה | שימוש בפרוטוקול אמין TCP על מנת להבטיח שהמידע שיועבר יגיע בצורה אמינה. | השתמשתי בפרוטוקול TCP, המידע מגיע באופן אמין כאשר אין שיבושים ברשת. |
| מכונה וירטואלית אינה מקבלת מספיק משאבים מהמערכת | המשאבים שניתנו למכונה הוירטואלית עליה רץ הפרויקט לא מספיקים, ולכן מתקשה המכונה לעבוד לפי צרכי הפרויקט | בינוני | נתינה של מספיק משאבים לכל מכונה וירטואלית (RAM, CPU,NETWORK,  STORAGE) | כאשר מריצים את התוכנה אני מקצה מספיק משאבים למכונה הוירטואלית (משתנה בין מחשב למחשב, לרוב המחשבים שנבדקו בשביל ריצה מלאה צריך 2 מעבדים ו2048MB RAM) |

# **תיאור ארכיטקטורה המערכת**

## **תיאור החומרה**

בפרויקט שלי ישנם רכיבים שונים שפועלים ביחד על מנת להביא את הפרויקט לעבוד בצורה מלאה ולאפשר למנהלים שונים לעקוב אחרי עבודת עובדיהם ועל אמינות עבודתם. ראשית נגדיר את מחשבי הלקוחות, אשר מהווים את אבני היסוד של הפרויקט. הלקוחות ישלחו את המידע אל מחשב השרת, כלומר אל מחשב המנהל.

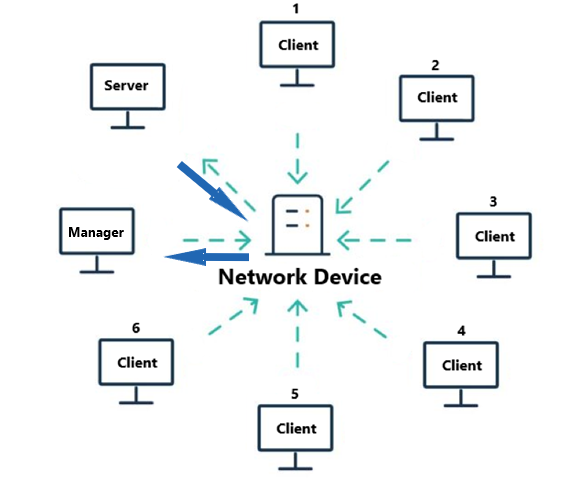
מחשב המנהל ימצא תחת אותה רשת עם מחשבי הלקוחות. המנהל יקבל את המידע מן הלקוחות דרך הרשת, המידע יועבר בצורה מוצפנת, הן בשימוש הצפנה אסימטרית להעברת המפתח והן בשימוש בהצפנה סימטרית בכדי להעביר מידע מוצפן עם המפתחות שהוחלפו בעזרת ההצפנה האסימטרית. מחשבי הלקוחות מתחברים לרשת באמצעות כרטיסי רשת המאפשרים להם לשלוח ולקבל נתונים בצורה מאובטחת.

חיבור הרשת בין הלקוח למנהל מתבצע דרך פרוטוקול TCP מאובטח, המבטיח שהנתונים יישלחו בצורה יציבה ומוגנת. מחשב השרת, שממוקם תחת אותה רשת, מקבל את המידע לפי הפרוטוקול המוסכם מהלקוחות ומעבד אותו בצורה מאובטחת. המחשב מצויד במערכת לניהול הצפנה המפענחת את המידע שהתקבל.

לאחר מכן, השרת בתקשורת עם המנהל ישלח לו בצורה מוצפנת את המידע כאשר המנהל מבקש. המנהל מציג את המידע בצורה גרפית בממשק משתמש נוח וברור, תוך שמירה על הצפנה ואבטחת המידע לאורך כל התהליך.

המנהל יכול לעקוב אחרי המידע בצורה ברורה ומסודרת. לצורך העברת המידע ברשת, ישנו גם תפקיד חשוב של רכיבי הרשת, כגון נתבים ומתגים, המוודאים שהמידע יגיע בצורה תקינה ממחשב הלקוח אל מחשב השרת והן ממחשב השרת אל מחשב המנהל ולהפך. הנתבים ומתגים אחראים על ניתוב המידע ומוודאים שהמידע יגיע ליעדו הנכון.

## **מראה גרפי של קשרי החומרה**



## **תיאור טכנולוגיות רלוונטיות**

### **שפות תכנות**

שפת תכנות C:

שפת C היא הבחירה המרכזית לפיתוח תוכנות ברמת ליבת מערכת ההפעלה. השפה מאפשרת גישה ישירה למשאבי המערכת ול-API של הליבה, מה שמעניק שליטה מלאה על תהליכים, זיכרון ורכיבי מערכת קריטיים. בעזרת C ניתן לבצע פעולות כמו עבודה עם מצביעים, ניהול זיכרון ישיר ותקשורת עם רכיבי חומרה או חלקים אחרים של המערכת.

שפת תכנות Python:

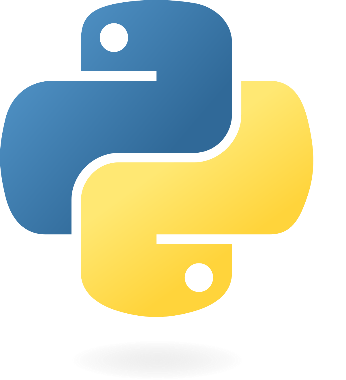
Python יכולה לשמש ככלי עזר לפיתוח סביבת העבודה של הפרויקט. היא אידיאלית לאוטומציה של בדיקות, איסוף נתונים ויצירת ממשקים לתקשורת עם רכיבי המערכת. Python מצטיינת בפשטות שלה וביכולות המתקדמות שלה בטיפול ברשתות, ניתוח קבצים ולוגים, ובניהול מהיר של משימות מורכבות.

שפת תכנות SQL:

בנוסף לשפות התכנות, SQLהיא כלי מרכזי לניהול ואחסון נתונים בצורה מסודרת. היא מאפשרת ליצור מסדי נתונים, לשמור מידע, ולעבוד איתו בקלות דרך שאילתות מובנות. בעזרת SQL ניתן לארגן מידע בטבלאות עם מבנה ברור, להגדיר קשרים בין נתונים שונים, ולשלוף בדיוק את המידע הדרוש בצורה מהירה ויעילה. השפה מתאימה לניהול כמויות גדולות של נתונים ומשמשת בסיס לרוב מערכות אחסון הנתונים המודרניות.

השילוב של C, Python ו-SQL נותן לי גמישות עצומה בעבודה שלי. כל אחת מהשפות מביאה יתרונות ייחודיים, והעבודה ביניהן מאפשרת לי ליצור מערכת חזקה ומותאמת אישית. שפת C מאפשרת לי לעבוד ישירות עם המערכת, עם שליטה מלאה על הזיכרון, התהליכים והמשאבים של המחשב. אני יכול לכתוב קוד יעיל שמבצע את הפעולות הקריטיות באופן מהיר ומדויק.

Python, לעומת זאת, נותנת לי דרך פשוטה ואינטואיטיבית ליצור סקריפטים שמנהלים תקשורת עם חלקי המערכת, מעבדים נתונים או מבצעים בדיקות. היכולת שלה לעבוד עם ספריות חזקות לרשתות ולעיבוד נתונים מקלה עליי לפתח חלקים מורכבים בצורה מהירה יותר. SQL

משתלבת כדי לאפשר לי לשמור נתונים בצורה מסודרת ולשלוף אותם בקלות לפי הצורך. במקום להתעסק עם אחסון נתונים גולמיים, אני יכול להשתמש במבנה של מסדי נתונים כדי לשמור על סדר ולייעל את העבודה שלי. השילוב בין השפות האלה מאפשר לי לעבוד עם שכבת הליבה, ולאחסן את המידע בצורה מאורגנת ונגישה. כל שפה תורמת לתפקיד שונה, וביחד הן יוצרות מערכת מאוזנת תמונה שמכילה גרפיקה, עיגול, צילום מסך, לוגו

התיאור נוצר באופן אוטומטיויעילה.

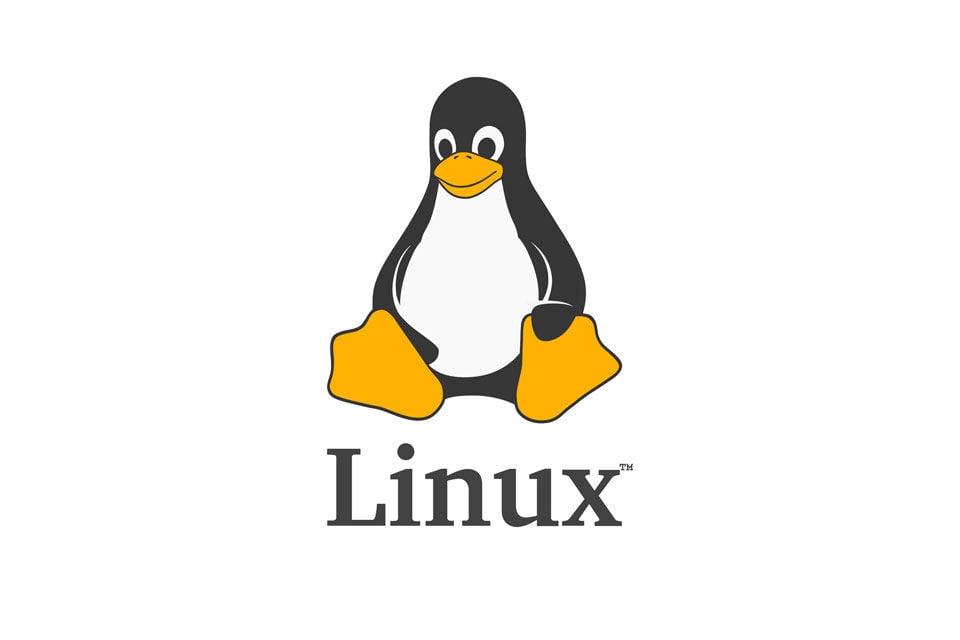
### **מערכת הפעלה**

לינוקס (Linux) היא מערכת הפעלה מבוססת קוד פתוח, כלומר, הקוד שלה זמין לציבור וניתן לשנות אותו, להפיץ אותו ולהשתמש בו ללא מגבלות מסחריות. זו אחת הסיבות לכך שלינוקס נפוצה מאוד בקרב מפתחים, חוקרי אבטחת מידע וסטודנטים שמחפשים ללמוד ולהתנסות במערכת הפעלה ברמה עמוקה.

לינוקס בנויה במודל מודולרי, כך שהרכיבים שלה מחולקים לגרעין ספריות, ותוכנות מערכת. הגרעין אחראי על אינטראקציה עם החומרה, ניהול תהליכים, זיכרון ומערכות קבצים. העובדה שהקוד של הגרעין זמין מאפשרת למפתחים לחקור אותו, להבין את אופן הפעולה של המערכת, ואפילו לכתוב תוספות או שינויים כמו דרייברים.

מכיוון שלינוקס ניתנת להתאמה אישית מלאה, קל יחסית ליצור פרויקטים כמו הפרויקט שלי. ניתן לשנות את קוד הגרעין כדי להוסיף פונקציונליות או להתקין מודול קרנל KLM שמבצע פעולות מתקדמות, כמו גישה לזיכרון, ניהול תהליכים, או עקיפה של בקרות גישה. בנוסף, קהילת המפתחים של לינוקס גדולה, ויש תיעוד רב שיכול לסייע בלמידה ובהתמודדות עם אתגרים טכניים.הגמישות והנגישות של לינוקס הופכות אותה לפלטפורמה אידיאלית לפרויקטים חינוכיים ומחקריים, במיוחד בתחום אבטחת המידע והסייבר.

מכונה וירטואלית היא סביבה שמדמה מחשב עצמאי בתוך מחשב פיזי. בעזרתה ניתן להריץ מערכת הפעלה (כמו לינוקס) בתוך מערכת הפעלה אחרת, כך שיש בידנו סביבה מבודדת לחלוטין, שמנצל את המשאבים של המחשב הפיזי כמו מעבד, זיכרון ואחסון, ומחלק אותם בין מכונות וירטואליות שונות. בפרויקטים שמצריכים עבודה על פיתוח ברמה נמוכה, כמו שינוי גרעין המערכת או עבודה עם מודולים קרנל, מכונה וירטואלית חשובה מאוד. היא מאפשרת ניסוי עם הקוד בסביבה מבודדת, כך שאין חשש לשבש את מערכת ההפעלה הראשית או את המחשב הפיזי. זה גם מאפשר ביצוע בדיקות וחקירות, תוך שמירה על הסביבה האמיתית מפני נזקים אפשריים.

**תקשורת**

התקשורת בפרויקט זה היא חלק מרכזי ומהותי בהצלחתו. המטרה היא לאפשר העברת מידע בין המערכת שבה רץ הקוד לבין מחשב אחר בצורה יעילה, אמינה ובטוחה. התקשורת מהווה את הגשר בין המידע שנאסף לבין היכולת לנתח ולהשתמש בו, ולכן תכנון נכון של מנגנון זה הוא חיוני. בפרויקט מתוכנן שימוש בפרוטוקול TCP (Transmission Control Protocol) לצורך התקשורת.

TCP נבחר בשל היתרונות המשמעותיים שלו באמינות ובניהול מסרים. זהו פרוטוקול שמבטיח שכל המידע הנשלח מגיע ליעדו בדיוק כפי שנשלח, ובסדר הנכון. אמינות זו קריטית במיוחד במערכת שמטרתה להעביר נתונים מדויקים. TCP פועל באמצעות יצירת חיבור מבוקר בין שני מחשבים, תהליך המתחיל ב"לחיצת יד משולשת" (Three-Way Handshake) שמבטיחה ששני הצדדים מוכנים לתקשורת. תהליך זה מאפשר גם התמודדות עם בעיות כמו אובדן נתונים או הפרעות בתקשורת, כיוון שהפרוטוקול כולל מנגנוני תיקון וידוא.

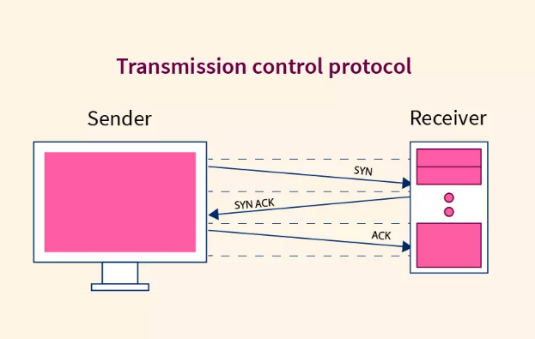
מרכיב חשוב נוסף בתקשורת בפרויקט הוא שימוש בנתונים מוצפנים. הצפנה היא כלי מרכזי להבטחת אבטחת המידע שנשלח, ובמיוחד כאשר מדובר בתקשורת שעשויה לכלול מידע רגיש או קריטי. ההצפנה מגנה על הנתונים מפני גישה לא מורשית, כך שגם אם צד שלישי מצליח ליירט את התקשורת, הוא לא יוכל לפענח את המידע שנשלח.

בעת תכנון הפרויקט, יינתן דגש לשילוב מנגנוני הצפנה חזקים, כמו שימוש באלגוריתמים סטנדרטיים כגון AES ו DH, כדי להבטיח רמת אבטחה גבוהה. הנתונים המוצפנים יישלחו במסגרת חיבור ה-TCP, כך שנוכל ליהנות גם מאמינות הפרוטוקול וגם מהגנה על המידע.

שילוב הצפנה בתקשורת מבטיח שהמערכת לא רק פועלת ביעילות אלא גם משמרת את אבטחת המידע. האיזון בין אמינות, יעילות ובטיחות הוא קריטי להצלחת הפרויקט, והתכנון המוקפד של התקשורת נועד להשיג מטרה זו בצורה מיטבית.

הפרויקט מתוכנן לפעול בתוך רשת מקומית. רשת מקומית היא מערכת תקשורת המחברת מספר מחשבים ומכשירים אחרים בתוך שטח גיאוגרפי מוגבל, כמו בית, משרד או מוסד חינוכי. ה-LAN מתאפיינת במהירות תקשורת גבוהה יחסית וביכולת לשלוט ברשת באופן ישיר, מה שהופך אותה לפתרון אידיאלי לפרויקט זה.

הבחירה לפעול בתוך רשת LAN נובעת ממספר סיבות: ראשית, העבודה ברשת מקומית מפחיתה את התלות באינטרנט החיצוני, מה שמאפשר גישה מהירה ואמינה יותר בין המחשבים ברשת. שנית, LAN מספקת רמה מסוימת של פרטיות, שכן התקשורת נשארת בתוך הרשת הפנימית ואינה נחשפת לרשתות חיצוניות שעלולות להיות פגיעות יותר להתקפות. בפרויקט זה, התקשורת תתבצע רק בין מחשבים הנמצאים באותה רשת מקומית, מה שמבטיח יעילות וביצועים מיטביים. הגבלה זו גם מפשטת את תהליך ההגדרה והניהול של התקשורת, כיוון שאין צורך להתמודד עם מורכבויות כמו ניתוב דרך כתובות IP חיצוניות או פתיחת פורטים בנתב. חשוב לציין כי בעוד שה-LAN מספקת יתרונות רבים מבחינת מהירות ואמינות, היא גם מציבה מגבלות – הפרויקט לא יוכל לתקשר עם מחשבים מחוץ לרשת המקומית. עם זאת, מגבלה זו היא חלק מהתכנון ומטרתה להבטיח פשטות, אבטחה ושליטה טובה יותר על המערכת.



### **תחומי עניין**

הפרויקט עוסק במגוון תחומי עניין טכנולוגיים המשלבים ידע תאורטי עם יישום מעשי. להלן תחומי העניין המרכזיים בפרויקט:

רשתות מחשבים ותקשורת נתונים

תכנון מנגנוני התקשורת בפרויקט נעשה תוך שימוש בפרוטוקול TCP, שמספק אמינות וניהול מסרים יעיל. נושא זה כולל הבנה מעמיקה של ניהול חיבורים, טיפול בהפרעות ושמירה על תקשורת תקינה בין רכיבי המערכת.

אבטחת מידע והצפנה

אחד ההיבטים המרכזיים בפרויקט הוא הגנה על הנתונים המועברים בזמן אמת באמצעות הצפנה. שילוב אלגוריתמים חזקים כמו AES ו DH מבטיח הגנה מפני גישה לא מורשית ושמירה על פרטיות המידע, גם אם הוא מיירט על ידי צד שלישי.

מערכות הפעלה וניהול משאבים

פיתוח מודול ליבה בלינוקס מחייב ידע מתקדם במבנה מערכת ההפעלה וניהול משאבים. התחום עוסק בשימוש יעיל במנגנונים כמו קריאות מערכת, ניהול זיכרון ותיאום בין תהליכים בזמן אמת.

תכנות ושפות מחשב

הפרויקט עושה שימוש בשפות כמו C, המאפשרות גישה ישירה למשאבים ושיפור ביצועים. תחום זה כולל כתיבת קוד ברמת ליבה תוך תשומת לב לאופטימיזציה ופתרון בעיות בזמן ריצה.

תכנון מערכות מורכבות

הפרויקט משלב בין מספר תחומי טכנולוגיה, מה שמחייב תכנון מקיף וחשיבה אסטרטגית. המערכת מתוכננת כך שתהיה יעילה, מאובטחת וקלה לתפעול, תוך שילוב פתרונות יצירתיים לאתגרים הנדסיים. תצוגת נתונים בזמן אמת

אחד ההיבטים החשובים בפרויקט הוא הצגת הנתונים הנאספים באופן גרפי ברור ואינטואיטיבי בזמן אמת, מבלי לשמור את המידע לאחר הריצה. המידע מוצג בממשק משתמש ידידותי, שמאפשר ניטור והבנה מיידיים של פעולת המערכת. תחום זה כולל שימוש בטכנולוגיות גרפיות ותכנון חוויית משתמש, תוך התחשבות על גמישות והתאמה למשתמשים.

תחומי העניין הללו מאפשרים ליצור פרויקט עשיר ומאתגר, שמחבר בין ידע טכנולוגי מתקדם לבין יישומים מעשיים בתחום מערכות ההפעלה, אבטחת המידע והנדסת התוכנה.

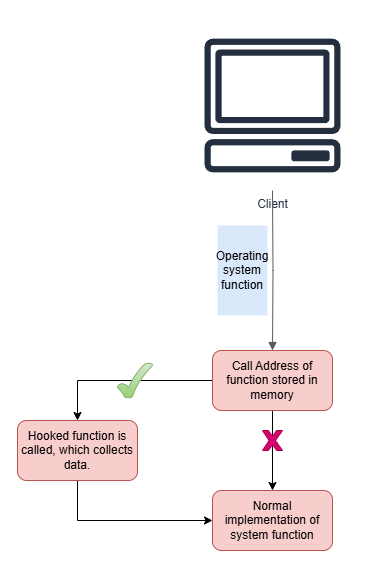
## **תיאור זרימת המידע במערכת**

### **האזנה שקטה על מחשב המשתמש**

תיאור הגרף:

גרף זה מציג את היכולת "האזנה שקטה על מחשב המשתמש". יכולת זו יכולה אך ורק לבוא לידי ביטוי על ידי שיתוף פעולה עם מערכת ההפעלה. ההאזנה השקטה מתבצעת באפשרות החבאת מידע מן המשתמש, שדבר זה אפשרי אך ורק בעזרת קוד מערכת הפעלה. דרך נגישה לעשות זאת מבלי לשנות את קוד מערכת ההפעלה עצמה (פעולה אפשרית, אך דורשת מכל מחשב שרוצה להריץ את פרויקט זה לשנות קוד במערכת ההפעלה שלו. כלומר פעולה ארוכה ולא יעילה, וגם מסוכנת אשר מדובר פה במערכת הפעלה, שהיא התוכנית הכי רגישה על המחשב, וכל שינוי מיותר בה יכול לגרור לסיכונים לכלל התנהלות המערכת) הינה על ידי השתמשות ב Kernel loadable module במערכת הפעלה לינוקס. לינוקס כמערכת הפעלה מאפשרת להוסיף אליה קוד אף תוך כדי זמן ריצה (מפה מגיעים המילים loadable module), הוספת אותו קוד הינו כלי חזק אשר התוכנית רצה בזיכרון של מערכת ההפעלה ובעלת גישה לנתונים רבים שאינם נגישים מהמשתמש. כפי שאפשר לראות בגרף, אפליקציות המשתמש יכולות רק לתקשר עם הקרנל, אך לא בעלות שינוי שלו. הוספת מודול משלנו למערכת הפעלה מאפשר לנו לשנות את הדרך ואת המידע שאליו אפליקציות המשתמש חשופות אליו.

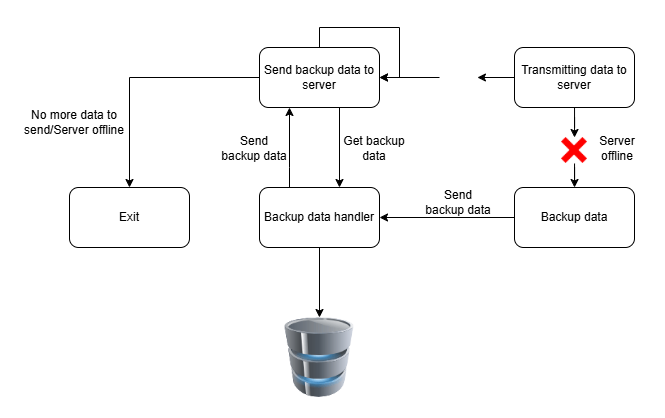
### **איסוף מידע בזמן אמת מן מחשב המשתמש**



תיאור הגרף:

גרף זה מציג את היכולת "איסוף מידע" במחשבי המשתמשים. איסוף המידע מתבצע באמצעות מונח הנקרא Hooking, מונח זה הינו תהליך שבו אנו יוצרים ניתוב מחדש של תהליך זרימת הקוד ולמעשה מנתבים אותו דרך פונקציות שלנו שבעזרתן נוכל להחליט לבצע האזנה בסתר על המידע המועבר במערכת הפעלה כחלק מקריאת הפונקציות. כמו שניתן לראות גם לאחר שינוי הפעולה בכדי שנוכל להאזין, הפעולה המרכזית עדיין נקראת, כלומר לא התבצעה פגיעה בפעולת מערכת, אלא רק האזנה מאחורי הקלעים.

### **גיבוי נתונים אצל המשתמש בעת שרת כבוי**

Check Mark Icon Green PNG Transparent Background, Free Download #45000 -  FreeIconsPNG

תיאור הגרף:

גרף זה מציג את היכולת "גיבוי נתונים" במחשבי המשתמשים. כאשר הלקוח מנסה לשלוח הודעה אל השרת והשרת איננו זמין (או כלשהי בעיה אחרת בשליחת ההודעה), הלקוח יגבה את המידע אצלו במחשב על ידי שימוש בממשק שיודע להתנהל כנגד מאגר הנתונים. כאשר הלקוח כן מצליח לתקשר עם השרת הוא יוודא כי אין מידע ששמור במאגר גיבוי המידע, ולכן ינסה לרוקנו כל פעם שיש חיבור מוצלח עם השרת.

### **שליחת מידע המשתמש אל המנהל**

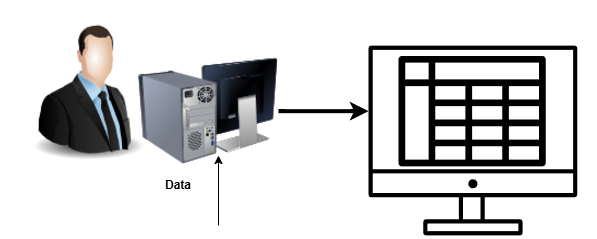
תמונה שמכילה צילום מסך, תרשים, עיצוב

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

תיאור הגרף:

גרף זה מציג את היכולת "שליחת מידע מהמשתמשים אל המנהל". השליחה מתבצעת על ידי איסוף המידע מהמחשבים של העובדים והעברתו דרך רכיבי אינטרנט. כל עובד מחובר דרך הרשת לרכיב אינטרנט, אשר מרכז את המידע מכל תחנות העבודה. לאחר מכן, המידע נשלח ישירות למחשב של השרת, שם הנתונים נאספים ועוברים עיבוד לצורך ניתוח או הפקת דוחות, לאחר מכן נשלח למנהל. פעולה זו מאפשרת לארגון לרכז את המידע במקום אחד ולהבטיח שהמנהל מקבל גישה למידע חיוני בזמן אמת.

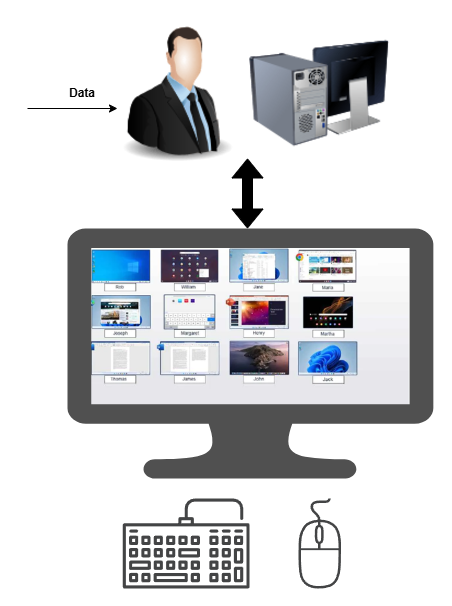
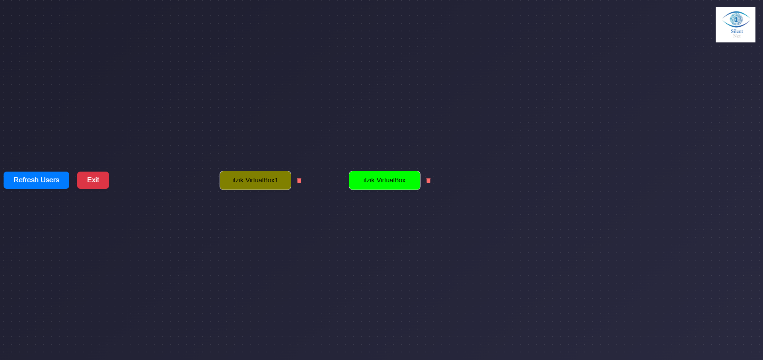
### **איסוף המידע במחשב השרת ל DB**



תיאור הגרף:

גרף זה מציג את היכולת "איסוף המידע במחשב השרת ל DB". איסוף המידע מתבצע על ידי שליחת נתונים מהמחשבים המקומיים של העובדים אל השרת, שם הוא שומר את המידע בצורה מסודרת בתוך DB, ה DB יפריד בין המידע שהוא מקבל מהמשתמשים, וכך יוכל המנהל להבחין בין סוגי המידע שנשלחים אליו. תהליך זה מתחיל באיסוף הנתונים, מהתוכנות או מהפעילות של המשתמשים במחשבים שלהם. הנתונים מועברים דרך רכיב אינטרנט באופן מאובטח, ולאחר מכן מגיעים לשרת המרכזי, שם הם עוברים עיבוד ראשוני ונשמרים בטבלאות מסודרות בבסיס הנתונים. תהליך זה מאפשר לארגון לנהל כמויות גדולות של מידע בצורה יעילה, להבטיח אחסון מאובטח של הנתונים, ולספק גישה מהירה לצורך ניתוח, קבלת החלטות או שיפור תהליכים עסקיים.

### **הצגת המידע באופן נגיש במחשב המנהל**



תיאור הגרף:

גרף זה מציג את היכולת "הצגת המידע באופן נגיש במחשב המנהל". תהליך זה מתאר את הדרך שבה הנתונים שנשמרו בשרת מוצגים בצורה נוחה וקלילה למשתמשים. המידע שנשמר בבסיס הנתונים במחשב השרת ונשלח אל המנהל אשר מציג בצורה מאורגנת על מסך, למשל כטבלה או גרף, כך שמנהלים יכולים לגשת אליו בקלות, להבין אותו בצורה מהירה ולבצע עיבודים נוספים אם נדרש. הנתונים המוצגים יכולים להיות תוצאות של חיפושים, דוחות או מידע אקטואלי הנוגע לפעילות המשתמש. הגישה למידע נעשית באמצעות מחשבים או מכשירים המחוברים לרשת, והמחשב המרכזי בשרת אחראי על שליחת המידע בצורה מסודרת, נגישה ובצורת המאפשרת למנהלים להבין את הנתונים במהירות וביעילות.

### Check Mark Icon Green PNG Transparent Background, Free Download #45000 - FreeIconsPNGCheck Mark Icon Green PNG Transparent Background, Free Download #45000 - FreeIconsPNGCheck Mark Icon Green PNG Transparent Background, Free Download #45000 - FreeIconsPNGCheck Mark Icon Green PNG Transparent Background, Free Download #45000 - FreeIconsPNG**הגדרת רמת בטיחות אצל המנהל**

תיאור הגרף:

גרף זה מציג את היכולת "הגדרת רמת בטיחות אצל המנהל". הגרף מתאר איך המנהל יכול להגדיר אצל השרת עד כמה לקוחות הוא מרשה ומכך השרת מגביל את כמות הלקוחות המחוברים אליו בו זמנית. בשל קושי ההמחשה בגרף, לא היה ניתן להמחיש את רמת הבטיחות שהמנהל יכול להגדיר, אך גם היא פרמטר שהמנהל מעביר לשרת בו הוא מחליט על כמות ההודעות הלא חוקיות שהשרת מרשה לכל לקוח ברצף. כפי שניתן לראות בגרף המנהל החליט כי כמות מקסימלית של לקוחות תהייה 4, ולכן רק ארבעה מחשבים הצליחו להתחבר אל השרת, בעוד השניים האחרים לא יכולים להתחבר לשרת ולא מקיימים session עימו.

### **ניסוח וניתוח של הבעיה האלגוריתמית**

הבעיה האלגוריתמית המרכזית בפרויקט שלי היא ניהול יעיל של אחסון הודעות, כך שלא תיגרם הצפה של המערכת במספר רב מדי של הודעות. הבעיה מתעוררת כאשר אין בקרה על כמות ההודעות, מה שעלול להוביל לשימוש מוגזם במשאבי מערכת כגון זיכרון או דיסק, ולגרום לקריסות או האטה משמעותית של התוכנה. דוגמה לכך היא כאשר הודעות נאגרות בקובץ במהירות גבוהה מהיכולת של המערכת לעבד ולמחוק אותן, וכתוצאה מכך גודל הקובץ תופס נפח רב ויוצר עומס.

### **אלגוריתמים קיימים לפתרון הבעיה**

הפתרון הנפוץ לבעיה זו הוא שימוש במבנה נתונים מעגלי (Circular Buffer), שבו האחסון מתנהל בצורה מעגלית: כאשר מגיעים לסוף המקום הפנוי, המצביע חוזר להתחלה ומתחיל לדרוס נתונים ישנים. כך, נשמרת שליטה על כמות הנתונים המאוחסנים, והמערכת לא מתמלאת ללא הגבלה.

קיימות גישות שונות לניהול Circular Buffer: למשל, שימוש בשני מצביעים (ראש ו-זנב), ניהול מונה הודעות, ובדיקת מצבי מלא/ריק. אפשר גם להגדיר מדיניות כמו דריסת הודעות ישנות אוטומטית או חסימת כתיבה כאשר אין מקום חדש עד שהתפנה מקום. לכל גישה יתרונות וחסרונות — דריסת נתונים ישנים מתאימה כאשר חשוב לשמור על זרימה רציפה, ואילו חסימה מתאימה כאשר לא רוצים לאבד מידע כלל.

### **סקירת הפיתרון הנבחר**

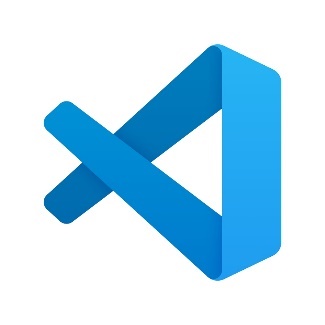
הפתרון שבחרתי — Circular Buffer בקובץ עם דריסה אוטומטית — מאפשר שמירה על ביצועים גבוהים ללא צורך בתחזוקה מורכבת. באמצעות ניהול פשוט של מצביעי קריאה וכתיבה, אני יכול לוודא שהקובץ לעולם לא יגדל מעבר לגבול מוגדר, וכך למנוע קריסה או האטת המערכת. בחרתי לא להשתמש בגישות אחרות כמו רשימות מקושרות או מבני תור מתוחכמים, כיוון שהן מוסיפות מורכבות מיותרת ואינן מתאימות לצורך הבסיסי של שליטה קלה וגמישה בגודל הנתונים. התממשקות עם קובץ שמתנהג כמקום אחסון מעגלי מביאה עימה את היכולת לאחסן בצורה יעילה ללא שימוש בפקודות שונות של מערכת ההפעלה הקשורות לקבצים. מימוש בצורה נכונה של אלגוריתם זה מייעל את הליכי הפרויקט ואף מבטיח על כמות מוגדרת מראש של מידע שלא תלך לאיבוד כאשר השרת לא פעיל אך הלקוח מבצע את עבודתו. מוטב לציין כי בפרויקט יש לשמור בתוך הקובץ עצמו את שני המצביעים של הקובץ, אשר יש להתחשב במקרה כאשר מחשב הלקוח קורס ויש מידע בתוך קובץ הגיבוי, לכן לאחר כל קריאה/כתיבה נעדכן את מצביעים אלו בסוף הקובץ.

### **הפנייה למקור רלוונטי**

[מהו באפר מעגלי](https://en.wikipedia.org/wiki/Circular_buffer)

## **תיאור סביבת הפיתוח**

תמונה שמכילה גרפיקה, עיצוב גרפי, לוגו, צילום מסך

התיאור נוצר באופן אוטומטיהסביבה בה אני עובד דורשת שימוש במכונה וירטואלית, משום שעלי להפעיל גרסה מסוימת של מערכת לינוקס, שהיא חשובה להרצת הקוד שלי. באמצעות מכונה וירטואלית, אני יכול ליצור סביבה מבודדת שמתפקדת בצורה עצמאית ומסייעת לי לבצע ניסויים ובדיקות בצורה בטוחה ומבוקרת. במכונה הווירטואלית אני מריץ גרסה מסוימת של לינוקס שיכולה להשתנות לפי הצורך, מבלי להשפיע על מערכת ההפעלה הראשית שלי. אני משתמש בכלים כמו VirtualBox או VMware כדי להקים ולהפעיל את הסביבה הווירטואלית הזו. לצורך פיתוח הקוד, אני משתמש בשני עורכי טקסט עיקריים: VIM ו-VSCode.VIM הוא עורך טקסט מאוד פופולרי ונוח למי שמכיר את הפקודות וזקוק לכלים בסיסיים, ומאפשר לי לכתוב קוד בצורה מהירה ויעילה, במיוחד בסביבת לינוקס. הוא קל, גמיש ומאפשר עבודה מהירה עם קבצים ללא צורך בממשק גרפי. מצד שני, אני משתמש גם ב-VSCode, שהוא עורך מודרני ומתקדם יותר, המציע אינטגרציה עם כלים נוספים כמו ניהול גרסאות, דיבאגינג, ותוספים רבים שמקליקים את העבודה עם קוד. בנוסף, אני משתמש ב-Git ו-GitHub לניהול גרסאות הקוד שלי. Git הוא כלי לניהול גרסאות שמאפשר לי לעקוב אחרי השינויים בקוד, לשמור על גרסאות קוד שונות ולשלוט בצורה יעילה בהתפתחויות הקוד לאורך הזמן. GitHub הוא שירות מבית Git שמאפשר לי לשתף את הקוד, לעבוד בצוות ולגבות את הקוד שלי. כלים אלו חשובים במיוחד כשיש צורך לעקוב אחרי שינויים רבים בקוד ולשתף את העבודה עם אחרים. כחלק מהתהליך, אני גם עושה שימוש ב-Wireshark כדי לנתח את התקשורת בין רכיבי המערכת. Wireshark הוא כלי מצוין לתפיסת וניתוח תעבורת רשת, אשר מאפשר לי לראות את זרימת המידע בין השרתים והלקוחות, ולוודא שאין בעיות בתקשורת או בשימוש בכתובת IP. זה חשוב במיוחד לבדיקת האינטראקציות וההעברות נתונים, כדי לוודא שהקוד עובד בצורה תקינה. לסיכום, אני עושה שימוש בסביבה מאוד עשירה שמספקת לי את כל הכלים הנדרשים לפיתוח, בדיקות, וניהול הקוד שלי. מכונה וירטואלית מספקת לי את הגמישות, עורכי הטקסט מאפשרים עבודה נוחה ומהירה, Git ו-GitHub חשובים לניהול הגרסאות, ו-Wireshark מאפשר ניתוח ופתרון בעיות בתקשורת. כלים אלו יחד יוצרים סביבה עוצמתית ומסודרת לפיתוח מתקדם.

## **תיאור פרוטוקול התקשורת**

החשיבות של פרוטוקול תקשורת בפרויקט שלי היא מכרעת, שכן הוא מבצע את התיאום וההבנה בין כל הרכיבים במערכת. פרוטוקול תקשורת מספק את הכללים וההנחות שדרכם רכיבי המערכת יכולים להבין אחד את השני ולהעביר מידע בצורה מסודרת וברורה. אם הפרוטוקול מוגדר בצורה טובה, הוא הופך את התקשורת לפשוטה יותר ומובנת לכל הצדדים המעורבים, ללא צורך בהבנה עמוקה של כל רכיב בנפרד. בפרויקט זה פרוטוקול התקשורת הינו פרוטוקול סינכרוני בין רכיבי התקשורת כאשר קיימת תלות בין סוגי ההודעות בתקשורת בין המנהל לשרת (בין העובד לשרת אין תלות).

מבנה הפרוטוקול תקשורת בו אשתמש בפרויקט הינו במבנה הבא:

1. אורך הודעה (כארבעה תווים שמהווים מספר)

2. סוג הפעולה שהתבצעה (המתודה שנקראה/איזה פעולה המשתמש ביצע)

3. המידע הקשור לאותה פעולה (פרמטרים)

התו המפריד להודעות יהיה הבית ‘\x1f’ (בזכות היותו בית שלא נמצא באף הודעה בין חלקי הפרויקט השונים, לכן לא יתערבב עם חלקי הודעה שונים). חשוב להתייחס לכך שאם התו המפריד יהיה חלק מהחלק השני של ההודעה, תיווצרנה בעיה בפרוטוקול, לכן נימנע מלשלוח הודעה שכוללת בחלק השני שלה את התו הנ"ל. לא תהיינה בעיה שהתו יופיע כחלק מהחלק השלישי של ההודעה, אשר ברגע שנמצא התו המפריד הראשון, שם נדע לבצע את ההפרדה בין חלקי ההודעה. כאשר יהיה שימוש בתו המפריד בחלק השלישי של ההודעה דבר זה יהיה בכדי לבצע הפרדה בין הפרמטרים של פקודה זו.

### **דוגמה להודעות:**

CIE\x1f480006 (שימוש באינפוט של חומרה)

CPO\x1fchrome.exe0014 (פתיחת תהליך).

ועוד...

### **טבלת הודעות פרוטוקול**

|  |  |  |  |
| --- | --- | --- | --- |
| שם ההודעה | מבנה שדות ההודעה | נשלח מ-/אל- | תיאור |
| CAU | תעודת זהות כרטיס רשת (MAC) ולאחר מכן שם הוסט (hostname) | לקוח אל שרת | הודעה הנשלחת כל תחילת תקשורת בין לקוח לשרת |
| CPO | שם תהליך | לקוח אל שרת | שם תהליך שנפתח על ידי הלקוח |
| CCU | מספרי ליבה ואחוזי שימוש בה מופרדים עם תו ההפרדה בין כל ליבה. לאחר הליבה האחרונה יופיע גם השעה בה נמדדו נתונים אלו –  0,56\x1f1,98,datetime | לקוח אל שרת | אחוזי שימוש בכל ליבה במעבד של המחשב |
| COT | קטגוריה של סוג אפליקציה | לקוח אל שרת | אפליקציות אליהן הלקוח מתקשר ברשת |
| CIE | מספר | לקוח אל שרת | קלט מכל חומרה המתקבל במחשב הלקוח מתקבל כקוד, אותו קוד נשלח לשרת |
| MME | אין (חוץ מסוג ההודעה ואורכה) | מנהל אל שרת | המנהל מודיע אל השרת על סיום החיבור ביניהם |
| MST | כמות לקוחות מקסימלית ולאחר מכן רמת בטיחות | מנהל אל שרת | לאחר התחברות המנהל אל השרת הוא ישלח לו את ההגדרות הרצויות למנהל – כמות לקוחות מקסימלית המחוברים לשרת ורמת בטיחות של השרת כנגד כל לקוח |
| MGC | אין (חוץ מסוג ההודעה ואורכה) | מנהל אל שרת | המנהל מבקש מן השרת את הנתונים הכלליים על הלקוחות המחוברים |
| MGD | שם הלקוח | מנהל אל שרת | המנהל מבקש את שלל הנתונים על לקוח ספציפי |
| MDG | שם הלקוח | מנהל אל שרת | המנהל שולח לשרת שם של לקוח שהוא רוצה למחוק את הנתונים שלו מהשרת |
| MCC | אין (חוץ מסוג ההודעה ואורכה) | מנהל אל שרת | המנהל בודק אם השרת פתוח לתקשורת, אחרת המנהל לא יהיה שימוש במערכת של המנהל |
| MCN | השם הנוכחי של הלקוח והשם החדש של הלקוח | מנהל אל שרת | המנהל מבקש להחליף שם של לקוח |
| MCH | אין (חוץ מסוג ההודעה ואורכה) | שרת אל מנהל | השרת מעדכן את המנהל כי עדכון השם הצליח |
| MMP | אין (חוץ מסוג ההודעה ואורכה) | מנהל אל שרת | המנהל מעדכן את השרת שהוא הולך לשלוח לו את הסיסמה |
| MIC | אין (חוץ מסוג ההודעה ואורכה) | שרת אל מנהל | השרת מעדכן את המנהל שהסיסמה ששלח לא נכונה |
| MVC | אין (חוץ מסוג ההודעה ואורכה) | שרת אל מנהל | השרת מעדכן את המנהל שהסיסמה ששלח תקפה והחיבור יוצא לפועל |
| EXH | ערכי DH – יכול להיות גם g/p וגם המפתח הציבורי | שרת/מנהל אל מנהל/שרת | החלפת מפתחות שמתחרשת בתחילת התקשורת בין השרת למנהל |

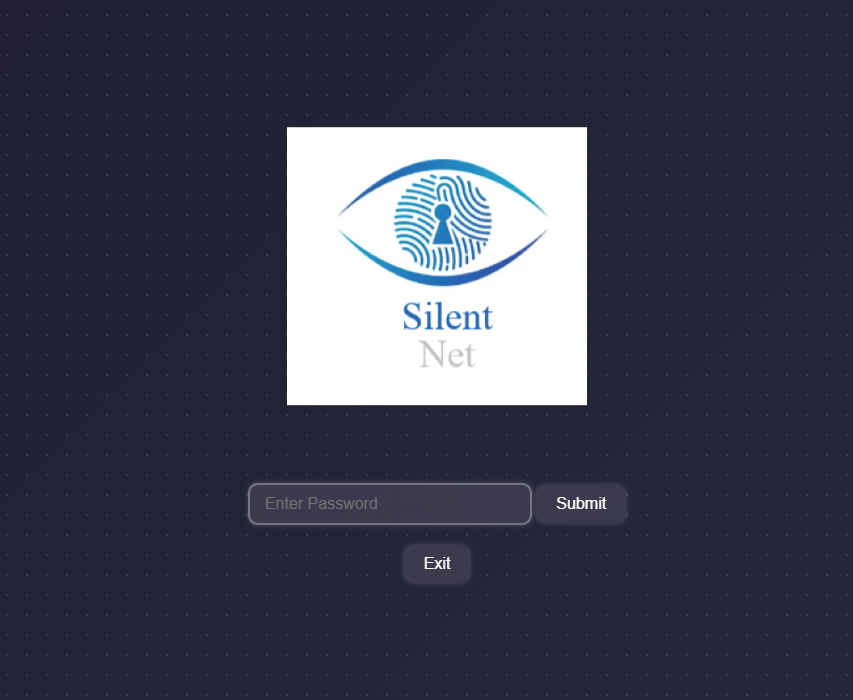
סוגי שגיאות:

MIH (שרת אל מנהל) - השרת מעדכן את המנהל כי עדכון השם לא חוקי .

MAC (שרת אל מנהל) - השרת מעדכן את המנהל כי קיים כבר מנהל מחובר לשרת ועל כך לא ייתן לעוד מנהל להתחבר

## **תיאור מסכי המערכת**

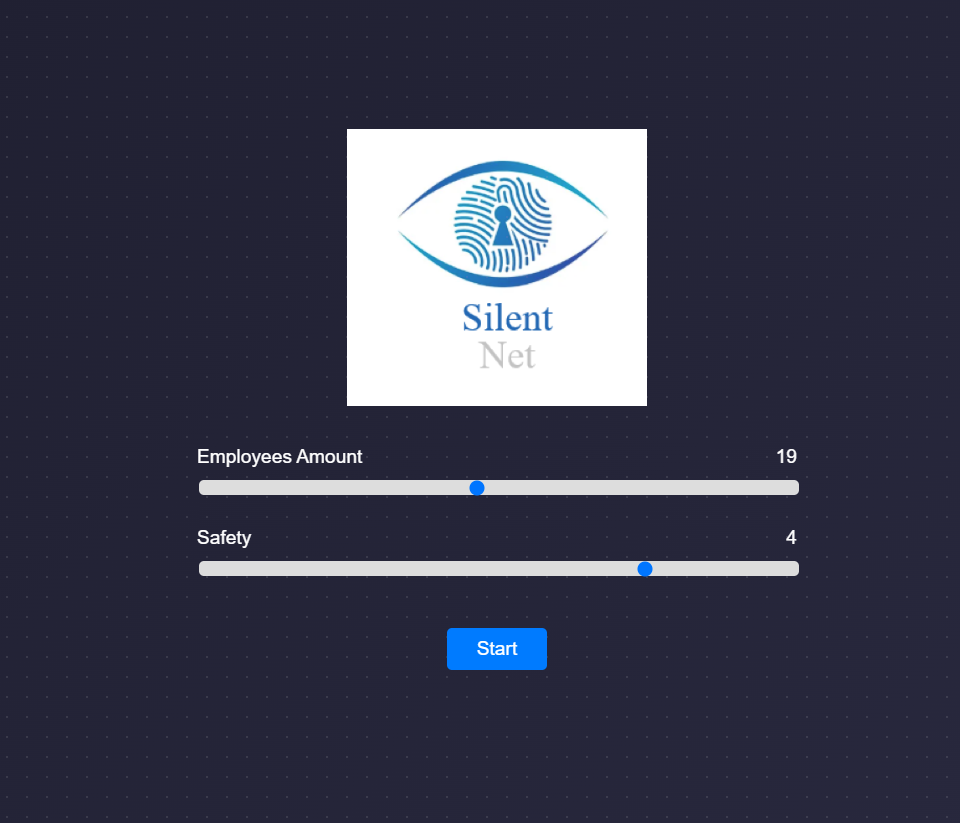
### **מסך פתיחה**



מהות מסך:

מסך זה הינו המסך הראשון ביותר שיופיע במחשב המנהל כאשר הוא פותח את התוכנית. מסך זה הינו מסך פתיחה, כלומר מחכה לקלט המשתמש על מנת להמשיך הלאה. כפי שאפשר לראות במסך זהו בעצם הלוגו של המערכת "Silent Net". במסך זה על המנהל להכניס סיסמה שידועה לו מראש, במידה והסיסמה נכונה ימשיך למסך הבא. במידה והסיסמה לא נכונה יופיע לו הודעה על כך ולא ימשיך למסך הבא. (במידה וכבר מחובר מנהל אחר, המנהל יופנה למסך הטעינה).

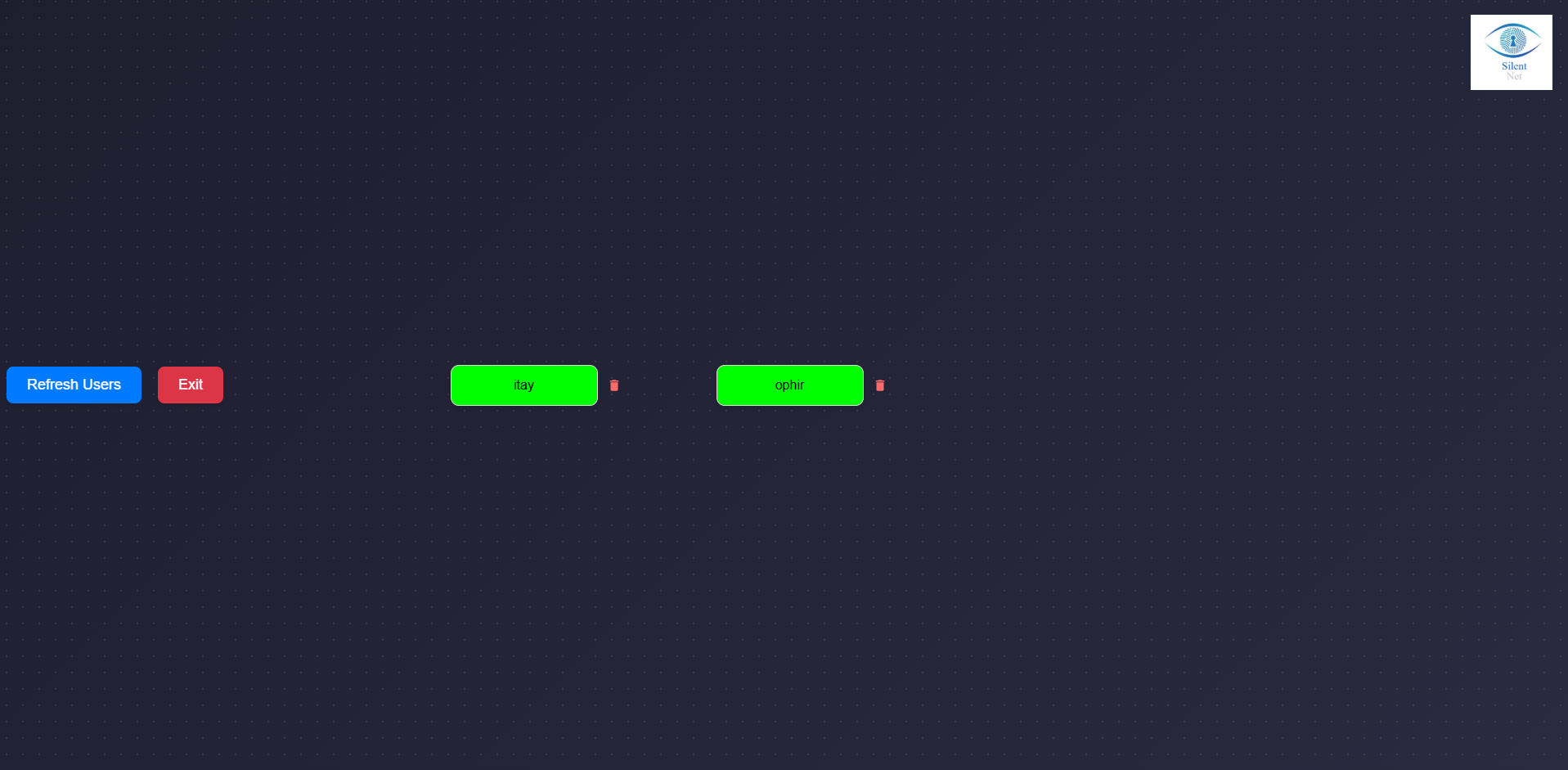
### **מסך הגדרות**



מהות מסך:

מסך זה הינו המסך השני אשר מוצג במחשב המנהל, מסך זה מבקש מן המנהל עצמו להכניס באופן ידני את ההגדרות הראשוניות ביותר לתוך הפרויקט. אחת מן ההגדרות (הראשונה) מבקשת את הכמות המקסימלית של לקוחות שיכולים להתחבר למחשב השרת. הגדרה זאתי מונעת מן מחשב השרת לקרוס בעת עומס, בנוסף על כך מונעת התקפה ידועה בשם DDOS/DOS בה מתחברים כמויות גדולות של לקוחות אל מחשב השרת במטרה להפיל אותו בשל עומס. ההגדרה השנייה קובעת את כמות הבטיחות של הפרויקט, כלומר הכוונה היותר מדויקת היא כמה מחשב השרת משאיר קשר עם מחשב מסוים, עד שהוא מנתק אותו בשל היות אותו קשר עם מחשב שמטרותיו זדוניות. רמת הבטיחות היא בסופו של דבר כמה מחשב השרת משאיר קשר עם מחשב לקוח לאחר שמחשב הלקוח שוב ושוב שלח מידע שלא ניתן לפיענוח ולא עומד בפרוטוקול. אם מחשב לקוח שולח באופן עקבי מידע לא אמין, רמת הבטיחות תקבע מתי כבר להפסיק להאמין לאותו לקוח ולנתק את הקשר איתו.

### **מסך ראשי**



מהות מסך:

מסך זה הינו המסך השלישי אשר מוצג במחשב המנהל, מסך זה מראה בזמן אמת את המחשבים המחוברים כרגע למחשב השרת ואשר מנהלים איתו קישור. אותם מחשבים שולחים מידע אל מחשב השרת. שמות המחשבים בדוגמה זו הינם שמות של משתמשים אמיתיים, אך בעת חיבור ראשוני שם המחשב יהיה כשם ה hostname (אשר אין שם אחר שייצג את מחשב זה והוא קריא).

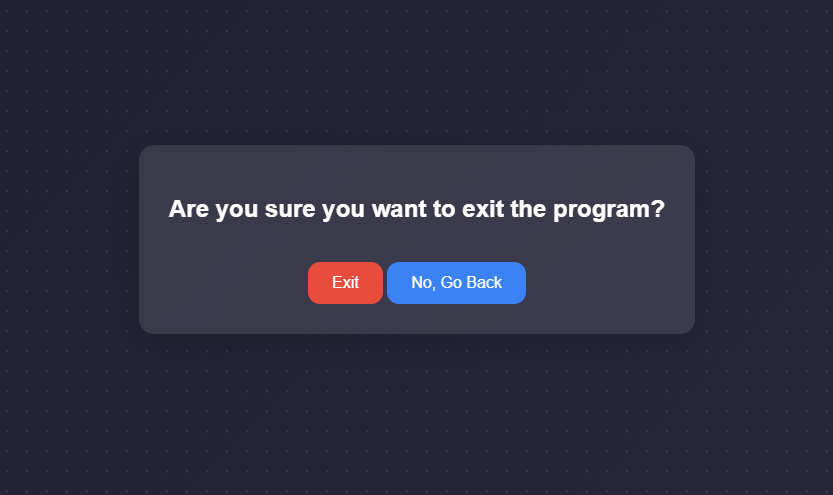
### **מסך אישי**



מהות מסך:

מסך זה הינו המסך הרביעי אשר מוצג במחשב המנהל. מסך זה מראה בזמן אמת את המידע אשר מתקבל מן אותו מחשב בעל כתובת מסוימת. ניתן להגיע למסך זה על ידי לחיצה על אחד השמות הקיימים במסך הראשי. ניתן לראות במסך את הנתונים השונים כמו תהליכים שנפתחו במחשב הלקוח, שימוש בליבות השונות, כמות מילים ממוצעת לדקה, זמנים בהם הלקוח לא היה פעיל, וסוגי IP אליהם פנה הלקוח

### **מסך יציאה**



מהות מסך

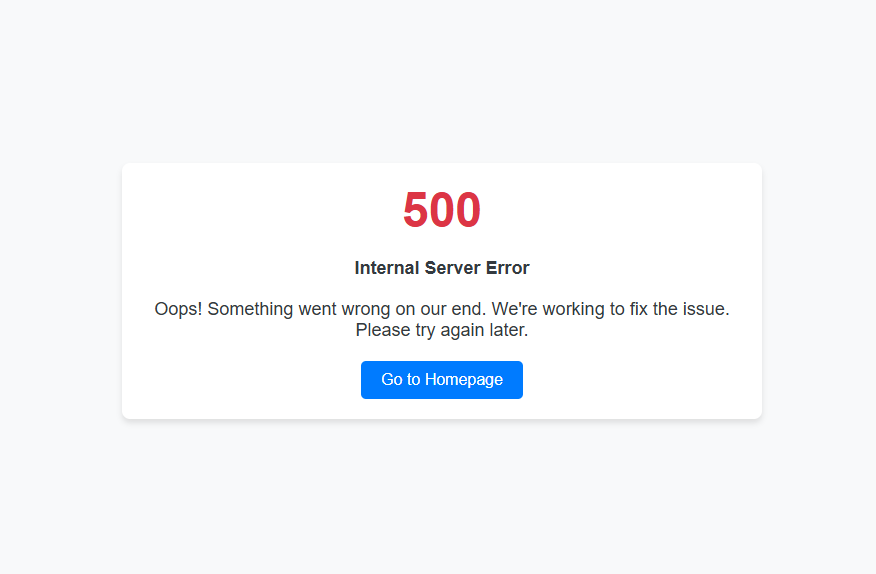
מסך יציאה, זהו המסך שיופיע למנהל לאחר שהמנהל רוצה לסגור את התוכנית. המסך בא לוודא שהמנהל לא רצה לצאת בטעות מן המערכת, ולכן שואל אותו שוב אם ברצונו לסגור ולצאת מן המערכת.

### תמונה שמכילה טקסט, צילום מסך, לוגו, עיצוב תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.**מסך טעינה**

מהות מסך:

מסך זה יופיע כאשר אבד החיבור עם השרת, ולכן המנהל יופנה למסך זה בו הוא יכול לנסות להתחבר אל השרת כאשר ירצה, אם החיבור יצליח המנהל יופנה למסך ההגדרות, אם לא, יישאר במסך זה.

### **מסך שגיאה**



מהות מסך:

מסך שגיאה, מופיע כאשר מתרחשת שגיאה אצל הקוד של התוכנית של המנהל. המסך מוביל למסך הבית שהוא מסך הטעינה.

### תמונה שמכילה טקסט, כרטיס ביקור, צילום מסך, גופן תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.**מסך 404**

מהות מסך:

כאשר המנהל ינסה לשנות את ה URL לכתובת לא חוקית (בתוך התוכנית), יופנה אל המנהל העמוד הזה שמוביל גם כן הוא אל מסך הבית – מסך הטעינה.

### **דיאגרמת מסכים**

תמונה שמכילה צילום מסך, טקסט, תרשים, עיצוב

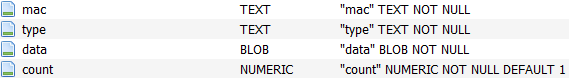
תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

## **תיאור מבני נתונים**

הפרויקט עוסק בשלל מבני נתונים שונים, המרכזיים ביניהם בהם עושה שימוש הם שתי טבלאות מרכזיות כמסד נתונים (בצד השרת) ובקובץ מרכזי (בצד הלקוח).

**צד שרת – מסד נתונים server\_db**

טבלה ראשונה - logs

תמונה שמכילה טקסט, צילום מסך, מספר, גופן

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.המידע שנשלח מהמחשב יכול לכלול נתונים שונים, קלט מחומרה, שמות תהליכים, ערכי CPU ועוד... כל מחשב שולח את המידע האחרון שנשלח, והמערכת בצד השרת מעדכנת באופן רציף בטבלה הראשונה. במערכת שלי, כל כתובת MAC משמשת כמפתח (key), וערך המפתח הוא המידע שנשלח מהמחשב הלקוח. המידע יכול להיות טקסטואלי (כמו שם תהליך), מספרי (כמו קוד של קלט מחומרה), או בייטים (כמו תו מפריד). כל עדכון מידע ממחשב ספציפי מעדכן את המספר של כמות הפעמים של אותו מידע. דוגמה למידע מתוך logs והגדרתו (אין מפתח בטבלה זו) -

השדות לפי התמונה משמאל: כתובת מאק הינו שדה מסוג טקסט וכך גם שדה סוג ההודעה, שדה המידע עצמו הוא שדה המוגדר כ BLOB – מידע בינארי. השדה האחרון הינו שדה שמונה את כמות הפעמים של ההודעה הנוכחית, וכמובן יהיה שדה מסוג מספר.

טבלה שנייה – uid

כל תקשורת בין לקוח לשרת מתחילה כאשר הלקוח שולח לשרת את שם ה hostname של המחשב וכתובת ה MAC של כרטיס רשת. מידע זה עוזר לשרת להבדיל בין הלקוחות השונים שמחוברים אליו. מידע זה נכתב בטבלה נפרדת בה רק שני הנתונים הללו שמורים, כתובת MAC ושם המחשב (שניהם מסוג טקסט). המידע נשמר בטבלה נפרדת בשביל הפרדה לוגית ובשל שוני ערכי הנתונים שצריך לשמור. הצורך בשמירת שם המחשב הוא בכדי שכאשר המנהל מבקש את רשימת המחשבים המחוברים הוא יוכל להסתכל עליהם לפי שמות ואף גם לשנות אותם לשמות אחרים, דבר שיקל עליו מאשר להסתכל על כל מחשב ככתובת ה MAC שלו. דוגמה למידע מתוך uid והגדרתו (השדות מוגדרים כ UNIQUE מה שמכריח את התוכנית לא להכניס את אותו MAC או hostname פעמיים – בדומה תמונה שמכילה טקסט, צילום מסך, גופן, קו

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.למפתחות) –

**צד לקוח – קובץ לגיבוי מידע**

בצד הלקוח, מיושם מנגנון גיבוי מקומי המיועד לשמירה זמנית של נתונים אשר לא עלה בידם להישלח אל השרת הייעודי עקב תקלות שרת או הפרעות בתקשורת הרשת. מנגנון זה ממומש באמצעות קובץ גיבוי בעל מבנה מעגלי (Ring Buffer) וגודל אחסון מוגדרת מראש.

בעת אירוע הדורש גיבוי, הלקוח מבצע העתקה מדויקת של המידע המיועד לשליחה אל קובץ זה, תוך שמירה על פורמט הנתונים המקורי כפי שהוא מוגדר בפרוטוקול התקשורת עם השרת. עם חידוש התקשורת התקינה עם השרת, הלקוח יוזם תהליך של שליחת הנתונים השמורים בקובץ הגיבוי אל השרת. מאחר וגודלו של קובץ הגיבוי הינו קבוע ומוגבל, מנגנון הדריסה המחזורית הינו חלק בלתי נפרד מפעולתו.

עם התמלאות הקובץ, נתונים חדשים יכתבו על גבי הנתונים הוותיקים ביותר, ובכך יאפשרו המשך פעולה רציפה של מנגנון הגיבוי גם תחת עומס נתונים מתמשך. השימוש בקובץ גיבוי בעל גודל קבוע הינו דרישה פונדמנטלית של הפרויקט. החלטה זו נובעת מן הצורך לצמצם את ההשפעה על משאבי המחשוב של הלקוח, ובפרט זיכרון ומשאבי דיסק.

באמצעות הגבלת גודל הקובץ, מתאפשרת פעולה יציבה וצפויה של מנגנון הגיבוי ברקע, באופן שקוף למשתמש הקצה וללא פגיעה בביצועי המערכת הכוללים. חשוב להדגיש כי המידע המאוחסן בקובץ הגיבוי משקף באופן ישיר את מבנה הנתונים המוגדר בפרוטוקול התקשורת בין הלקוח לשרת.

הקובץ אינו מבצע כל שינוי או עיבוד של המידע, ותפקידו היחיד הוא שימור זמני של נתונים טקסטואליים או בינאריים המיועדים לשליחה עתידית. ארכיטקטורה זו מבטיחה נאמנות לנתונים המקוריים ומפשטת את תהליך השחזור והשליחה מחדש בעת חידוש הקישוריות עם השרת.

## **סקירת חולשות ואיומים**

בפרויקט זה העוסק במגוון נושאים מעולם הסייבר אחד הנושאים שצריך לקחת בחשבון הוא ייצור קוד בטוח אשר יודע להתגונן נגד מתקפות שונות בכדי להבטיח ריצה חלקה ומלאה של הפרויקט ללא הפרעות ומבלי חשש לחדירה לנתונים פרטיים. הפרויקט נמנע מאיומים שונים ומתגונן בפני חולשות בנושאים השונים ובדרכים הבאות –

שכבת האפליקציה

Web

בפרויקט קיימת עבודה עם Web לצורך הצגה נוחה של GUI לצד המנהל. בכדי להימנע ממתקפות שונות הפרויקט נמנע מלתת למנהל (אשר הוא הרכיב היחידי בפרויקט בעל GUI) גישה להכנסת קלט כצורת טקסט (בכדי להימנע ממתקפות דומות למתקפה XSS – מתקפה ידועה שמנצלת את העובדה שאתרים לא בודקים או מנקים נכון קלט של משתמש, ומאפשרים להכניס קוד ישירות לדף(.

בנוסף, מתקפה ידועה נוספת היא כאשר האתר אינו בודק כראוי ואוכף את הגישה לקבצים שונים, כלומר המשתמש יכול לפי ה URL לשנות אותו ולהגיע לקבצים שלא היו בכוונת כותב השרת ציבוריים. הפרויקט נמנע ממתקפה כזאת על ידי שהוא נותן גישה אך ורק ל URLים ספציפיים, ואם השם המבוקש לא נמצא בהם שגיאה 404 תופיע למנהל.

אומנם הפרויקט נמנע מלתת גישה לקלט טקסט מן המשתמש ב GUI אך כן יש מקרים בודדים בהם קיים שימוש והם כאשר המנהל מתבקש להכניס סיסמה בכדי להיכנס למערכת וכאשר המנהל רוצה להחליף שם לאחד המחשבים המחוברים. הסכנה פה היא מתקפת - sql injection טכניקת תקיפה שבה תוקף מזריק פקודות SQL זדוניות לשדה קלט במטרה לשנות את שאילתת המסד נתונים ולבצע פעולות לא מורשות.

כאשר מכניסים סיסמה אין אפשרות לבצע מתקפה אשר בצד השרת בשלב זה עוד לא ניגשים אל מסד הנתונים. אך כאשר מחליפים שם ניתן לבצע מתקפה ולפגוע במסד הנתונים, אך הפרויקט נמנע ממתקפה זו על ידי בדיקת תווים מיוחדים בשמות שניתנו, אם המנהל נתן שם עם תווים אלו שיכולים לבצע פגיעה, הפרויקט לא ישלח את השם אל השרת.

תהליך login – צד לקוח וצד שרת

צד שרת:

כחלק מהליך ה login המנהל צריך לשלוח סיסמה שרק היא ידועה למנהלים, כלומר לא כל אדם יכול להשיג את הנתונים שהשרת מספק למנהלים, אלא רק מי שיודע את הסיסמה איתה השרת התחיל.

צד לקוח:

כחלק מהליך ה login הלקוח צריך לשלוח את הנתונים שיעזרו לשרת לזהות את לקוח זה – כתובת ה MAC של כרטיס הרשת וה - hostname של המחשב.

שני תהליכי login אלו קריטיים אשר מנהל/לקוח שלא יבצע אותם כמו שצריך לא ימשיך ב session עם השרת, כלומר לא ניתן לשלוח הודעות לשרת שמדמות הודעות שנשלחות באמצע session ולהשיג מידע/לשלוח מידע (שישמר בשרת) מבלי לבצע את התהליך ההתחלתי. הדבר מונע התחזות לאנשים.

הצפנה

כמו שצוין בחלק הקודם, המנהל שולח סיסמה אל השרת בתחילת ה session כחלק מתהליך ה login שלו. כמצופה, לא ניתן לשלוח סיסמה מבלי להצפין אותה אחרת האזנה פשוטה על ידי מתקפה MITM (Man in the middle – מחשב שמאזין למידע הנשלח בין שני מחשבים אחרים מבלי ידיעתם) יכולה לגרום להפצת הסיסמה ואנשים שהם לא המנהל יוכלו להשיג את המידע ולמחוק אותו!

בפרויקט נעשה שימוש בהצפנה שמתנהלת בין המנהל לשרת (עוד לפני שליחת הסיסמה). ההצפנה מונעת מכל מחשב שמאזין להבין את המידע הנשלח בין מחשב המנהל למחשב השרת ולא יהיה לו תועלת מכך. סוגי ההצפנות ופרוטוקולים שהפרויקט משתמש בהם הן:

DH (Diffie Hellman) – פרוטוקול להחלפת מפתחות ידוע שמסתמך על קושי פתרון בעיית הדיסקרט לוגריתם (Discrete Logarithm Problem), הפרוטוקול מאפשר לשני אנשים להגיע למפתח פרטי על ערוץ ציבורי (כך שאם מישהו מאזין לתקשורת הוא אינו יכול להגיע למפתח הפרטי). הפרויקט משתמש במספרים גדולים וחזקים שיקחו זמן רב בשביל מחשב מאזין לפענח בכדי להגיע למפתח הפרטי.

לאחר ששני הצדדים בעלי המפתח הפרטי ניתן כעת להשתמש בהצפנה המידע עם אותו מפתח פרטי שמשותף רק לשניהם. בשלב הזה הפרויקט משתמש בהצפנה סימטרית AES (מצב CBC) – אלגוריתם להצפנה סימטרית שפועל במצב שרשור בלוקים (Cipher Block Chaining) כדי לשפר את האבטחה על ידי קישור כל בלוק מוצפן לבלוק הקודם, הצפנה חזקה שמקשה על מי שינסה להאזין לתעבורה.

חשוב לציין כי יש חשיבות גדולה לשימוש ב TCP עם מצב CBC בהצפנה AES, אשר מכיוון שההצפנה הינה הצפנת בלוקים, כל בלוק מסתמך על הבלוק הקודם כחלק מההצפנה, ולכן גם הקידוד יעבוד באותה צורה, כלומר עם בלוק אחד הולך לאיבוד או אפילו רק חלק ממנו, דבר שמאוד יתכן בפרוטוקולים כמו UDP, לא יהיה ניתן לפענח את המסרים מהצד השני.

DOS/DDOS

אחת המתקפות הקלות לביצוע אך האפקטיביות מבניהן. ההתקפה מסתמכת ישירות על כך שמשאבים של מחשב אינם אינסופיים, ולכן אם רכיב אחד מקבל כמות עצומה של משימות בבת אחת הוא ככל הנראה יקרוס/לא יתפקד כמצופה. הפרויקט נמנע ממתקפה זו בכמה דרכים שונות:

* הגבלת כמות הלקוחות אצל השרת, השרת פשוט לא ייצור session עם לקוחות כאשר כבר קיימים כמות לקוחות מחוברים ככמות שהמנהל הגביל את השרת. בנוסף המנהל יכול גם כן להגביל את השרת לכמות בעצמה מוגבלת של לקוחות (המנהל לא יכול לתת מעל 40 לקוחות, כמות גדולה אך מעליה השרת מתקשה בביצוע פעילותו).
* הגבלת כמות המנהלים שיכולים להיות מחוברים לשרת בו זמנית. בפרויקט רק מנהל אחד יכול לנהל session עם השרת (אפילו שהמנהל מקבל יחס עליון – גם אם כמות הלקוחות הגיעה לכמות הלקוחות המקסימלית המנהל עדיין יכול להתחבר).
* הגדרת רמת בטיחות ללקוחות. המנהל יכול להגדיר לשרת את רמת הבטיחות שהוא רוצה כאשר הוא מחובר לשרת, רמת הבטיחות היא מקושרת ישירות לכמות ההודעות שהשרת לא יכול לפענח שמגיעה מן לקוחות, לאחר כמות ההודעות האלו שמגיעה ברצף השרת סוגר את ה session עם הלקוח. הגדרה זו מונעת מלקוח שכן התחבר כראוי לשרת לבצע מתקפת DOS על השרת.

שכבת התעבורה – פרוטוקול TCP

TCP מספק חיבור אמין (reliable) בין שני צדדים באמצעות מנגנון של לחיצת יד משולשת (Three-Way Handshake). במהלך לחיצת היד, שני הצדדים מאמתים אחד את השני ומסכימים על פרמטרים בסיסיים של החיבור. התהליך מבטיח שהחיבור נוצר בכוונה תחילה ולא נוצר בטעות או בהתקפת זיוף פשוטה (כגון IP spoofing). בנוסף, TCP כולל מנגנוני בקרת שגיאות - לכל מקטע (segment) נוסף Checksum — מספר שמייצג את תוכן המקטע.

כשמקבלים מקטע, בודקים אם החישוב מתאים. אם יש טעות, המקטע נזרק והצד השולח ישלח אותו שוב, סידור מנות - לפעמים מנות מגיעות בסדר שונה מזה שבו הן נשלחו (בגלל רשתות). TCP נותן לכל מקטע מספר סידורי (Sequence Number), וכשהצד המקבל מקבל מנות, הוא מסדר אותן לפי המספרים כדי לשחזר את הנתונים בדיוק כמו שנשלחו, ובקרת זרימה - TCP מוודא שהשולח לא יציף את המקבל ביותר מדי נתונים.

לשם כך המקבל מודיע לשולח כמה מקום פנוי יש לו בזיכרון (בחלון שנקרא Window Size). כך, השולח מתאים את קצב השליחה למה שהמקבל מסוגל לקלוט ולעבד, שמגינים על שלמות הנתונים ומקטינים סיכון לפגיעות כתוצאה מהעברת נתונים חלקית או שגויה.

Timing attack

סוג של תקיפת צד-ערוץ שבה התוקף מנתח את הזמן שלוקח למערכת לבצע פעולות שונות, כדי לגלות מידע סודי כמו סיסמאות, מתקפה זו מסתמכת על ההתנהגות הטבעית של המחשב בביצוע פעולות שונות. אם צד אחד יודע את הדרך בה מתבצעת פעולה מסוימת הוא יכול למדוד זמנים כמה זמן לוקח לכל קלט להחזיר תשובה מהשרת.

הדרך שבה השרת בודק אם הסיסמה נכונה היא על ידי השוואת אות אות, ברגע שאות לא טועמת את האות בסיסמה השמורה, הוא מסיים את הבדיקה ומחזיר FALSE, עם ידע זה, הצד המתקיף יכול לנסות סטרינגים שונים ולראות מה לוקח לשרת יותר זמן להגיב, ולפי זה הוא יכול לדעת אם הוא בכיוון לסיסמה הנכונה לא.

ניתן לבצע מתקפה זו אם המנהל בטעות השאיר את המחשב שלו פתוח ועובד השיג גישה למחשב המנהל (בהסתמך שהתוכנית סגורה כאשר המנהל לא על המחשב, אם המנהל לא סוגר את התוכנית כשהוא לא על המחשב ועובדים יכולים להשיג גישה למחשב שלו זה כבר מעבר לתחום הפרויקט), העובד יכול לנסות לשלוח סיסמאות שונות בדף פתיחה אל השרת ולראות מה לוקח יותר זמן.

הפתרון לבעיה זו הוא דווקא פשוט למימוש בצד השרת – לאחר כל פעם שהשרת מאמת סיסמה שנשלחה אליו הוא מחכה למשך זמן רנדומלי (עד לשנייה) ורק לאחר מכן מגיב בחזרה לצד השני. ביצוע פעולה זו גורם לכך שאין באפשרות הצד השני לעולם לדעת אם הסיסמה מקיימת בתוכה חלק מתוך הסיסמה האמיתית ובכך לא יקבל רמזים לגבי הסיסמה האמיתית.

# **מימוש הפרויקט**

## **מודולים/מחלקות מיובאים**

מנהל

sys - משמש להוספת תיקיית האב ל־sys.path כדי לאפשר ייבוא של קבצים משותפים כמו protocol, encryption.

webbrowser - פותח את ממשק הניהול בדפדפן ברירת המחדל כאשר האפליקציה מופעלת.

os - משמש להרצת פעולות מערכת כמו קבלת נתיב קובץ נוכחי ושליחת אות סיום לתהליך (os.kill).

signal - משמש לשליחת אותSIGINT כדי לעצור את התהליך בצורה נקייה בעת יציאה מהמערכת.

Json - משמש לפענוח מידע בפורמט JSON שמתקבל מהשרת, למשל נתוני סטטיסטיקה של עובדים.

functools.wraps - דקורטור שעוזר לשמור על המידע של הפונקציה המקורית כאשר עוטפים אותה בפונקציה אחרת (כמו בדקורטור check\_screen\_access).

Flask מחלקת Flask) – משמשת ליצירת אפליקציית ווב בפלאסק).

redirect (פונקציה) – מחזירה תגובת הפניה לכתובת אחרת (למשל לאחר התחברות או שגיאה).

render\_template (פונקציה) – מציגה דף HTML מתוך תיקיית התבניות (templates) עם אפשרות לשלוח נתונים לדף.

request (אובייקט) – מייצג את הבקשה שנשלחה על ידי המשתמש, כולל מידע כמו פרמטרים, טפסים ו־JSON.

jsonify (פונקציה) – ממירה מבני נתונים של פייתון (כמו dict אוlist ) לתגובה בפורמט JSON.

url\_for (פונקציה) – יוצרת כתובת URL לפונקציה לפי שמה, בצורה בטוחה ודינמית.

שרת

sys – משמש לקבלת פרמטרים מהשורה (כמו מספר לקוחות וסיסמה) ולהוספת תיקיות ל sys.path

threading – משמש לניהול חיבורים מרובים בו־זמנית באמצעות יצירת תהליכונים (threads) לכל לקוח.

os – משמש לגישה למערכת הקבצים, קבלת נתיבים ל־DB, ופעולות מערכת כלליות.

json – משמש להמרה של מידע ל־JSON כדי לשלוח אותו למנהל דרך הפרוטוקול.

time.sleep – משמש להשהיית פעולה לזמן רנדומלי כדי למנוע התקפות תזמון (timing attacks)

random.uniform – מגריל מספרים עשרוניים לטווח מסוים, משמש גם כן למניעת התקפות תזמון.

keyboard.on\_press\_key – מקשיב ללחיצות מקשים (q, e) להפסקת השרת או למחיקת הלוגים.

socket.timeout – מטפל במצבים שבהם מתבצעת קריאה לחיבור שחרג מהזמן שהוקצה לו.

traceback – משמש להדפסת מידע מפורט על שגיאות שהתגלו במהלך ריצת התוכנית לצורכי ניפוי שגיאות (debugging).

לקוח

types – מספק טיפוסי נתונים כמו uint32\_t ו-int64\_t לניהול נתונים יעיל בליבה.

uaccess – מאפשר גישה מאובטחת לזיכרון משתמש להעתקת נתונים בין מרחבי כתובות.

fs – מספק פונקציות לניהול קבצים במערכת, כולל קריאה, כתיבה וסגירה.

init – מאפשר הגדרת פונקציות אתחול (\_\_init) וניקוי (\_\_exit) עבור המודול.

input – מספק מבנים ופונקציות לטיפול באירועי קלט ממקלדת ועכבר.

kernel – מכיל פונקציות ליבה בסיסיות כמו printk לניפוי שגיאות.

kernel\_stat – מספק גישה לסטטיסטיקות מעבד, כגון זמני פעילות ומנוחה.

kprobes – מאפשר הוספת נקודות תצפית (hooks) לפונקציות ליבה.

module – הכרחי להגדרת מודול ליבה, כולל רישוי, מחבר ותיאור.

mutex – מספק נעילות לסנכרון תהליכים במצבים מרובי־תהליכונים.

netdevice – מאפשר גישה לרשימת התקני רשת וכתובות MAC

sched – מספק גישה למבנה current לזיהוי תהליכים ומשתמשים.

slab – מספק פונקציות להקצאת ושחרור זיכרון בליבה.

utsname – מאפשר קבלת שם המחשב (hostname) לזיהוי המערכת.

workqueue – מאפשר ניהול משימות רקע (background tasks) בצורה אסינכרונית.

ftrace – משמש להסתרת חיבורי רשת על ידי שינוי התנהגות פונקציות ליבה.

in – מכיל פונקציות כמו in\_aton להמרת כתובות IP

inet– מספק פונקציות רשת כמו htons להמרת פורטים.

net – מכיל הגדרות בסיסיות לחיבורי רשת ומבני נתונים כמו struct sock

ptrace – מאפשר גישה לרגיסטרים במהלך הוקים של פונקציות ליבה.

seq\_file – משמש להסתרת חיבורי רשת בעת הצגתם בפקודות כמו netstat

tcp – מספק פונקציות ומידע על חיבורי TCP לניהול תקשורת.

dcache – מאפשר גישה למבני נתונים של מערכת הקבצים לניהול קבצים נסתרים.

err – מכיל קודי שגיאה סטנדרטיים לטיפול בתקלות.

mount – מספק פונקציות לניהול נקודות עגינה (mount points) במערכת הקבצים.

namei – מאפשר פתיחה וניהול של קבצים באמצעות מסלולים paths

stat – מכיל הגדרות הרשאות ופונקציות לבדיקת תכונות קבצים.

## **מודולים/מחלקות מקוריים**

### **manager.py**

Class: SilentNetManager

**תפקיד:** מחלקה זו היא הלב של אפליקציית האינטרנט בצד המנהל. היא משתמשת ב- Flask כדי ליצור את ממשק המשתמש, לטפל בניתוב ולתקשר עם השרת. היא מנהלת את המסכים השונים של האפליקציה (כניסה, הגדרות וכו') ואת הזרימה ביניהם.

**תכונות:**

* + app (Flask) מופע אפליקציית Flask.
  + manager\_socket (client או None): אובייקט socket לתקשורת עם השרת.
  + is\_connected (bool) דגל המציין אם המנהל מחובר לשרת.
  + screens (dict) מילון הממפה נתיבי URL לרמות היררכיית מסך.
  + current\_screen (str) ה-URL של המסך המוצג כעת.
  + previous\_screen (str) ה-URL של המסך שהוצג קודם לכן.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מאתחל את אפליקציית הניהול, מגדיר את אפליקציית Flask, השקע, היררכיית המסכים והנתיבים.
* \_setup\_routes:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מגדיר את כל הנתיבים של Flask עבור אפליקציית האינטרנט.
* \_setup\_error\_handlers:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מגדיר מטפלי שגיאות עבור שגיאות HTTP 404 ו-500.
* check\_screen\_access:
  + טענת כניסה: פונקציית תצוגה ניתנת לקריאה של Flask (f).
  + טענת יציאה: פונקציה עטופה ניתנת לקריאה שאוכפת היררכיית מסכים.
  + תיאור: דקורטור שבודק אם למשתמש יש הרשאה לגשת למסך בהתבסס על היררכיה מוגדרת.
* disconnect:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מתנתק מהשרת ומנקה את השקע.
* exit\_program:
  + טענת כניסה: אין
  + טענת יציאה: תגובה ריקה עם קוד סטטוס 204.
  + תיאור: מטפל ביציאה מהאפליקציה, מתנתק מהשרת ומסיים את התהליך.
* page\_not\_found:
  + טענת כניסה: אובייקט שגיאה (בדרך כלל 404).
  + טענת יציאה: תבנית HTML מעובדת עבור דף השגיאה 404.
  + תיאור: מעבד את דף השגיאה 404.
* internal\_error:
  + טענת כניסה: אובייקט שגיאה (בדרך כלל 500).
  + טענת יציאה: תבנית HTML מעובדת עבור דף השגיאה 500.
  + תיאור: מעבד את דף שגיאת השרת הפנימית 500.
* exit\_page:
  + טענת כניסה: אין
  + טענת יציאה: תבנית HTML מעובדת עבור מסך אישור היציאה.
  + תיאור: מעבד את דף אישור היציאה.
* loading\_screen:
  + טענת כניסה: אין
  + טענת יציאה: תבנית HTML מעובדת עבור מסך הטעינה.
  + תיאור: מעבד את מסך הטעינה בזמן ניסיון להתחבר לשרת.
* start\_screen:
  + טענת כניסה: אין
  + טענת יציאה: תבנית HTML מעובדת עבור מסך הכניסה הראשוני, פוטנציאלית עם דגל המציין סיסמה שגויה.
  + תיאור: מעבד את מסך הכניסה הראשוני.
* check\_password:
  + טענת כניסה: סיסמה מטופס הכניסה.
  + טענת יציאה: תגובת הפניה, או למסך ההגדרות (בכניסה מוצלחת), למסך הטעינה (אם החיבור נכשל), או למסך הפתיחה (אם הסיסמה שגויה).
  + תיאור: מאמת את סיסמת המנהל מול השרת.
* settings\_screen:
  + טענת כניסה: אין
  + טענת יציאה: תבנית HTML מעובדת עבור מסך הגדרות השרת.
  + תיאור: מעבד את מסך הגדרות השרת.
* submit\_settings:
  + טענת כניסה: נתוני טופס המכילים את מספר העובדים והגדרות הבטיחות.
  + טענת יציאה: תגובת הפניה למסך העובדים.
  + תיאור: מעדכן את הגדרות השרת עם הערכים שסופקו.
* employees\_screen:
  + טענת כניסה: אין
  + טענת יציאה: תבנית HTML מעובדת עבור מסך רשימת העובדים, המכילה רשימה של סטטיסטיקות עובדים.
  + תיאור: מעבד את מסך רשימת העובדים, ושולף נתונים מהשרת.
* delete\_client:
  + טענת כניסה: נתוני JSON המכילים את שם הלקוח למחיקה.
  + טענת יציאה: תגובת JSON המציינת הצלחה או כישלון של המחיקה.
  + תיאור: מטפל בבקשה למחיקת לקוח מהשרת.
* manual\_connect:
  + טענת כניסה: אין
  + טענת יציאה: תגובת JSON המציינת את סטטוס החיבור.
  + תיאור: מנסה להתחבר לשרת באופן ידני ומחזיר את סטטוס החיבור.
* update\_client\_name:
  + טענת כניסה: נתוני JSON המכילים את השם הישן והשם החדש של הלקוח.
  + טענת יציאה: תגובת JSON המציינת הצלחה או כישלון של עדכון השם.
  + תיאור: מטפל בבקשה לעדכון שם של לקוח.
* stats\_screen:
  + טענת כניסה: שם של לקוח.
  + טענת יציאה: תבנית HTML מעובדת עבור מסך הסטטיסטיקות המפורטות, המכילה נתוני סטטיסטיקה של הלקוח.
  + תיאור: מעבד את מסך הסטטיסטיקות המפורטות עבור לקוח מסוים, ושולף נתונים מהשרת.
* connect\_to\_server:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מנסה להתחבר לשרת.
* run:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מריץ את אפליקציית Flask.

### **server.py**

Class: SilentNetServer

**תפקיד:** זוהי אפליקציית ליבת השרת. היא מנהלת את כל חיבורי הלקוח והמנהל, מקבלת ומעבדת נתונים, ויוצרת אינטראקציה עם מסד הנתונים. היא אחראית לפעולה הכוללת של מערכת "Silent Net" .

**תכונות:**

* max\_clients (int) המספר המרבי של לקוחות עובדים המורשים להתחבר.
* safety (int) סף בטיחות לטיפול בהודעות לא חוקיות.
* password (str) הסיסמה הנדרשת לאימות מנהל.
* proj\_run (bool) דגל המציין אם השרת פועל.
* manager\_connected (bool) דגל המציין אם מנהל מחובר כעת.
* clients\_connected (list) רשימה של טאפלים, שכל אחד מהם מכיל אובייקט ת'רד ואובייקט לקוח עבור לקוחות עובדים מחוברים.
* macs\_connected (list) רשימה של כתובות MAC של לקוחות עובדים מחוברים.
* clients\_recv\_event (threading.Event) אירוע המשמש לסנכרון קבלת נתוני לקוח.
* log\_data\_base (UserLogsORM) מופע של המחלקה UserLogsORM לניהול לוגים.
* uid\_data\_base (UserId) מופע של המחלקה UserId לניהול זהויות משתמשים.
* server\_comm (server) אובייקט שרת לתקשורת רשת.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מאתחל את השרת עם הגדרות ברירת מחדל.
* start:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מפעיל את השרת עם ההגדרות שהוגדרו.
* \_load\_configuration:
  + טענת כניסה: אין או ארגומנטים משורת הפקודה (מספר לקוחות מקסימלי, רמת בטיחות וסיסמה).
  + טענת יציאה: אין
  + תיאור: טוען את הגדרות השרת משורת הפקודה או משתמש בברירות מחדל.
* \_initialize\_databases:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מאתחל חיבורים למסדי הנתונים.
* \_setup\_keyboard\_shortcuts:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מגדיר קיצורי מקלדת לשליטה בשרת.
* \_run\_server:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: לולאת השרת הראשית לקבלה וטיפול בחיבורי לקוחות.
* \_accept\_clients:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מקבל ומנהל חיבורי לקוחות נכנסים.
* \_handle\_client\_connection:
  + טענת כניסה: אובייקט לקוח.
  + טענת יציאה: אין
  + תיאור: קובע את סוג הלקוח ומנתב למטפל המתאים.
* \_determine\_client\_type:
  + טענת כניסה: אובייקט לקוח, סוג הודעה, תוכן הודעה.
  + טענת יציאה: ערך בוליאני: True אם הלקוח היה מנהל, False אחרת.
  + תיאור: קובע אם הלקוח הוא מנהל או עובד ומטפל בהתאם.
* \_handle\_manager\_connection:
  + טענת כניסה: אובייקט לקוח, הודעת סיסמה.
  + טענת יציאה: ערך בוליאני: תמיד מחזיר True (חיבור מנהל מטופל).
  + תיאור: מטפל באימות וחיבור של מנהל.
* \_handle\_employee\_connection:
  + טענת כניסה: אובייקט לקוח, הודעת אימות.
  + טענת יציאה: אין
  + תיאור: מטפל באימות וחיבור של עובד.
* \_remove\_disconnected\_client:
  + טענת כניסה: אובייקט לקוח להסרה.
  + טענת יציאה: אין
  + תיאור: מסיר לקוח מנותק מרשימת הלקוחות המחוברים.
* quit\_server:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מסיים את פעולת השרת.
* erase\_all\_logs:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: מוחק את כל הלוגים ממסד הנתונים.

Class: ManagerHandler

**תפקיד:** מחלקה זו, הפועלת בצד השרת, מטפלת בתקשורת ספציפית מלקוח מנהל מחובר. היא מקבלת בקשות מהמנהל (למשל, עבור נתוני לקוח, מחיקת לקוח) ומעבדת אותן, לעתים קרובות באינטראקציה עם מסד הנתונים.

**תכונות:**

* server (SilentNetServer): הפניה למופע השרת הראשי.
* client (client): חיבור ה-socket למנהל.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה: מופע של SilentNetServer, אובייקט לקוח.
  + טענת יציאה: אין
  + תיאור: מאתחל את המטפל במנהל עם השרת ואובייקט הלקוח המתאים.
* process\_requests:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: לולאה ראשית לעיבוד בקשות ממנהל.
* \_handle\_client\_request:
  + טענת כניסה: סוג הודעה, תוכן הודעה.
  + טענת יציאה: תוכן תגובה.
  + תיאור: מנתב בקשות לקוח לפונקציות טיפול מתאימות.
* \_get\_client\_data:
  + טענת כניסה: שם לקוח.
  + טענת יציאה: נתוני סטטיסטיקה של לקוח.
  + תיאור: אוסף נתוני סטטיסטיקה מפורטים עבור לקוח מסוים.
* \_handle\_name\_change:
  + טענת כניסה: פרמטרי הודעה (שם קודם ושם חדש).
  + טענת יציאה: סוג הודעה המציין הצלחה או כישלון.
  + תיאור: מטפל בבקשה לשינוי שם לקוח.
* \_delete\_client:
  + טענת כניסה: פרמטרי הודעה (שם לקוח).
  + טענת יציאה: אין
  + תיאור: מטפל בבקשה למחיקת לקוח.
* \_handle\_unsafe\_message:
  + טענת כניסה: אין
  + טענת יציאה: ערך בוליאני המציין אם לנתק את המנהל.
  + תיאור: מטפל בהודעות לא בטוחות/לא חוקיות מהמנהל.

Class: ClientHandler

**תפקיד:** מחלקה זו, הפועלת בצד השרת, מטפלת בתקשורת עם לקוח (עובד). היא מקבלת נתונים מהלקוח (כגון שימוש במעבד, תהליכים פתוחים, הקלדות) ומעבדת אותם, ומאחסנת את המידע הרלוונטי במסד הנתונים.

**תכונות:**

* server (SilentNetServer): הפניה למופע השרת הראשי.
* client (client): חיבור ה-socket ללקוח.
* mac\_address (str): כתובת ה-MAC של הלקוח המשויך למטפל זה.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה: מופע של SilentNetServer, אובייקט לקוח, כתובת MAC של לקוח.
  + טענת יציאה: אין
  + תיאור: מאתחל את המטפל בלקוח עם השרת, אובייקט הלקוח וכתובת ה-MAC המתאימה.
* process\_data:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: לולאה ראשית לעיבוד נתונים מלקוח.
* \_handle\_client\_data:
  + טענת כניסה: סוג נתונים, נתוני הודעה.
  + טענת יציאה: אין
  + תיאור: מנתב נתוני לקוח לפונקציות טיפול מתאימות.
* \_handle\_ip\_data:
  + טענת כניסה: נתוני הודעה (כתובות IP).
  + טענת יציאה: אין
  + תיאור: מטפל בנתוני כתובות IP מהלקוח.
* \_handle\_process\_data:
  + טענת כניסה: נתוני הודעה (שמות תהליכים).
  + טענת יציאה: אין
  + תיאור: מטפל בנתוני שמות תהליכים מהלקוח.
* \_handle\_cpu\_data:
  + טענת כניסה: נתוני הודעה (נתוני שימוש במעבד).
  + טענת יציאה: אין
  + תיאור: מטפל בנתוני שימוש במעבד מהלקוח.
* \_handle\_keyboard\_data:
  + טענת כניסה: נתוני הודעה (אירועי לוח מקשים).
  + טענת יציאה: אין
  + תיאור: מטפל בנתוני אירועי לוח מקשים מהלקוח.

### **DB.py**

Class: DBHandler

**תפקיד:** מחלקת בסיס לטיפול בפעולות מסד נתונים. היא מספקת את הבסיס לחיבור, שאילתות וניהול מסד נתונים SQLite. היא נועדה להיות מחלקה ממנה יורשות מחלקות אחרות שצריכות גישה למסד נתונים.

**תכונות**:

* DB\_NAME (str): קבוע ברמת המחלקה המגדיר את שם קובץ מסד הנתונים ("server\_db.db").
* conn (sqlite3.Connection): משתנה מופע המייצג את החיבור למסד הנתונים SQLite.
* cursor (sqlite3.Cursor): משתנה מופע המייצג את אובייקט הסמן המשמש לביצוע שאילתות SQL.
* table\_name (str): משתנה מופע המאחסן את שם הטבלה הראשית שהמטפל משויך אליה.
* \_lock (threading.Lock): נעילת ת'רד כדי להבטיח פעולות מסד נתונים בטוחות לת'רד.

**פעולות:**

* init:
  + טענת כניסה: אובייקט חיבור למסד נתונים, אובייקט סמן למסד נתונים, שם הטבלה.
  + טענת יציאה: אין.
  + תיאור: מאתחל חיבור למסד נתונים באמצעות אובייקטי חיבור וסמן קיימים.
* connect\_DB:
  + טענת כניסה: שם מסד הנתונים.
  + טענת יציאה: טאפל המכיל אובייקט חיבור ואובייקט סמן.
  + תיאור: מקים חיבור למסד נתונים ומחזיר אובייקט חיבור וסמן.
* close\_DB:
  + טענת כניסה: אובייקט סמן, אובייקט חיבור.
  + טענת יציאה: אין.
  + תיאור: סוגר את החיבור למסד הנתונים.
* clean\_deleted\_records\_DB:
  + טענת כניסה: אין.
  + טענת יציאה: אין.
  + תיאור: מנקה את כל הרשומות שנמחקו מהטבלה.
* delete\_all\_records\_DB:
  + טענת כניסה: אין.
  + טענת יציאה: אין.
  + תיאור: מוחק את כל הרשומות מהטבלה ומנקה את מסד הנתונים.
* commit:
  + טענת כניסה: פקודת SQL, ארגומנטים לפקודה (אופציונלי).
  + טענת יציאה: תוצאות השאילתה (ייתכן מבנה נתונים מורכב בהתאם לשאילתה).
  + תיאור: מבצע פקודת SQL במסד הנתונים עם נעילה להבטחת סנכרון.

Class: UserLogsORM

**תפקיד:** מחלקה זו (בדומה ל-ORM) מרחיבה את DBHandler כדי לנהל רישום של נתוני פעילות משתמש. היא מספקת שיטות להוספת לוגים, אחזור סטטיסטיקות (כגון WPM, שימוש במעבד) וניתוח התנהגות משתמש. היא משתמשת בתבנית Singleton כדי להבטיח שקיים רק מופע אחד. **תכונות:**

* יורשת conn, cursor, table\_name, \_lock , מ - DBHandler

**פעולות:**

* new:
  + טענת כניסה: המחלקה עצמה (cls), אובייקט חיבור למסד נתונים, אובייקט סמן למסד הנתונים, שם הטבלה.
  + טענת יציאה: מופע סינגלטון של המחלקה.
  + תיאור: מבטיח שתיווצר רק מופע יחיד של המחלקה ומאתחל אותו עם חיבור וסמן קיימים.
* client\_setup\_db:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: אין.
  + תיאור: כותב לוגים בסיסיים שצריכים להיות עבור כל לקוח בעת התחברות.
* delete\_mac\_records\_DB:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: אין.
  + תיאור: מוחק את כל הרשומות מהטבלה של כתובת MAC מסוימת.
* \_\_check\_inactive:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: טאפל המכיל את הזמן האחרון שהלקוח היה פעיל, ואת מספר הדקות שהוא לא פעיל.
  + תיאור: בודק אם הלקוח כרגע לא פעיל.
* \_\_update\_last\_input:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: אין.
  + תיאור: מעדכן את הזמן האחרון שהמשתמש ביצע אירוע קלט.
* \_\_get\_total\_active\_time:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: מספר דקות הפעילות הכוללות.
  + תיאור: מחשב את זמן הפעילות הכולל של המשתמש.
* \_\_update\_cpu\_usage:
  + טענת כניסה: כתובת MAC של הלקוח, נתוני שימוש במעבד.
  + טענת יציאה: אין.
  + תיאור: מעדכן את רשומות ניצול המעבד.
* insert\_data:
  + טענת כניסה: כתובת MAC של הלקוח, סוג הנתונים, הנתונים עצמם.
  + טענת יציאה: אין.
  + תיאור: מכניס נתונים לטבלת SQL, אם הרשומה כבר קיימת מגדיל את המונה שלה.
* get\_process\_count:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: רשימת טאפלים של שם התהליך ומספר הפעמים שנפתח.
  + תיאור: מקבל את מספר הפעמים שכל תהליך נפתח עבור לקוח מסוים.
* get\_inactive\_times:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: רשימת טאפלים של זמן ומשך חוסר פעילות.
  + תיאור: מחשב את זמני חוסר הפעילות של המשתמש.
* get\_wpm:
  + טענת כניסה: כתובת MAC של הלקוח, רשימת זמני חוסר פעילות, אינדיקציה אם היה חוסר פעילות אחרי האירוע האחרון.
  + טענת יציאה: מספר מילים לדקה ממוצע.
  + תיאור: מחשב את ממוצע המילים לדקה שהמשתמש מקליד תוך התעלמות מזמנים לא פעילים.
* get\_cpu\_usage:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: טאפל של מילון ליבות מעבד וניצולן ורשימת זמני לוגים.
  + תיאור: מקבל את כל הלוגים של ניצול המעבד.
* get\_active\_precentage:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: אחוז הזמן שהמשתמש היה פעיל.
  + תיאור: מחשב את אחוז הזמן שהמשתמש היה פעיל.
* get\_reached\_out\_ips:
  + טענת כניסה: כתובת MAC של הלקוח.
  + טענת יציאה: רשימת כתובות IP שהמשתמש פנה אליהן.
  + תיאור: מקבל את כל כתובות ה-IP שלקוח מסוים פנה אליהן.

Class: UserId

**תפקיד:** מחלקה זו גם מרחיבה את DBHandler ומשתמשת בתבנית Singleton. היא אחראית על ניהול זיהוי משתמשים במערכת. היא מאחסנת ומאחזרת מידע משתמשים על סמך כתובות MAC ושמות מארחים.

**תכונות:**

* יורשת conn, cursor, table\_name, \_lock , מ – DBHandler

**פעולות:**

* new:
  + טענת כניסה: המחלקה עצמה (cls), אובייקט חיבור למסד נתונים, אובייקט סמן למסד הנתונים, שם הטבלה.
  + טענת יציאה: מופע סינגלטון של המחלקה.
  + תיאור: מבטיח שתיווצר רק מופע יחיד של המחלקה ומאתחל אותו עם חיבור וסמן קיימים.
* delete\_mac:
  + טענת כניסה: כתובת MAC של ההתקן.
  + טענת יציאה: אין.
  + תיאור: מוחק כתובת MAC מסוימת מהטבלה.
* insert\_data:
  + טענת כניסה: כתובת MAC של ההתקן, שם המארח של ההתקן.
  + טענת יציאה: ערך בוליאני המציין אם המשתמש כבר מחובר.
  + תיאור: מכניס נתונים לטבלת SQL, בודק אם ה-MAC כבר בשימוש או אם שם המארח קיים.
* update\_name:
  + טענת כניסה: שם המארח הקודם, שם המארח החדש.
  + טענת יציאה: אין.
  + תיאור: מנהל משנה שם ללקוח.
* check\_user\_existence:
  + טענת כניסה: שם המארח של ההתקן.
  + טענת יציאה: ערך בוליאני (או מספר שלם המציין קיום).
  + תיאור: בודק אם לקוח מסוים כבר מחובר.
* get\_clients:
  + טענת כניסה: אין.
  + טענת יציאה: רשימת טאפלים של MAC ושם מארח.
  + תיאור: מקבל את כל הנתונים על לקוחות MAC ושם מארח.
* get\_mac\_by\_hostname:
  + טענת כניסה: שם המארח של ההתקן.
  + טענת יציאה: כתובת MAC.
  + תיאור: מקבל את כתובת ה-MAC המתאימה של מחשב לפי שם המארח.

### **encryption.py**

Class:DiffieHellman

**תפקיד:** מחלקה זו מטפלת בחילופי מפתחות Diffie-Hellman. היא מאפשרת לשני צדדים להסכים על סוד משותף מעל תווך לא מאובטח.

**תכונות:**

* prime (int): מספר ראשוני המשמש לחילופי המפתחות.
* base (int): מספר הבסיס המשמש לחילופי המפתחות.
* private\_key (int): מפתח פרטי שנוצר באופן אקראי עבור צד זה.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה:
    - prime (int): מספר ראשוני לחילופי המפתחות.
    - base (int): מספר בסיס לחילופי המפתחות.
  + טענת יציאה: אין
  + תיאור: מאתחל את חילופי Diffie-Hellman עם מספר ראשוני ובסיס. אם prime או base הם 0, הוא מייצר פרמטרים חדשים.
* \_generate\_private\_key:
  + טענת כניסה: אין
  + טענת יציאה: מפתח פרטי (int)
  + תיאור: מייצר מפתח פרטי אקראי.
* get\_public\_key:
  + טענת כניסה: אין
  + טענת יציאה: מפתח ציבורי (int)
  + תיאור: מחשב ומחזיר את המפתח הציבורי.
* get\_shared\_secret:
  + טענת כניסה:
    - other\_public\_key (int): המפתח הציבורי של הצד השני.
  + טענת יציאה: סוד משותף (int)
  + תיאור: מחשב את הסוד המשותף באמצעות המפתח הציבורי של הצד השני.

Class: AESHandler

**תפקיד:** מחלקה זו מטפלת בהצפנה ופענוח של AES במצב CBC. היא מספקת שיטות להצפנת ופענוח נתונים באמצעות מפתח AES.

**תכונות:**

* key (bytes, אופציונלי): מפתח ה-AES המשמש להצפנה ופענוח. אם לא סופק, נוצר מפתח אקראי.
* cipher (AES): אובייקט הצפנה של AES.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה:
    - key (bytes, אופציונלי): מפתח AES.
  + טענת יציאה: אין
  + תיאור: מאתחל את AESHandler עם מפתח. אם לא ניתן מפתח, נוצר מפתח אקראי.
* encrypt:
  + טענת כניסה:
    - data (bytes או str): הנתונים להצפנה.
  + טענת יציאה: נתונים מוצפנים (bytes)
  + תיאור: מצפין נתונים באמצעות AES במצב CBC.
* decrypt:
  + טענת כניסה:
    - encrypted\_data (bytes): הנתונים המוצפנים לפענוח.
  + טענת יציאה: נתונים מפוענחים (bytes)
  + תיאור: מפענח נתונים מוצפנים באמצעות AES במצב CBC.

Class: EncryptionHandler

**תפקיד:** מחלקה זו משלבת את חילופי Diffie-Hellman ואת הצפנת AES כדי לספק תקשורת מוצפנת. היא מטפלת בחילופי המפתחות הראשוניים ולאחר מכן משתמשת ב-AES להצפנת ופענוח הודעות.

**תכונות:**

* dh (DiffieHellman) : מופע של המחלקה DiffieHellman לטיפול בחילופי המפתחות.
* aes\_handler (AESHandler, אופציונלי): מופע של המחלקה AESHandler לטיפול בהצפנת AES לאחר השגת סוד משותף.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה:
    - prime (int): מספר ראשוני לחילופי Diffie-Hellman.
    - base (int): בסיס לחילופי Diffie-Hellman.
  + טענת יציאה: אין
  + תיאור: מאתחל את EncryptionHandler עם פרמטרים של Diffie-Hellman.
* generate\_shared\_secret:
  + טענת כניסה:
    - other\_public\_key (int): המפתח הציבורי של הצד השני.
  + טענת יציאה: אין
  + תיאור: מייצר את הסוד המשותף ומאתחל את AESHandler עם מפתח נגזר.
* encrypt:
  + טענת כניסה:
    - data (bytes או str): הנתונים להצפנה.
  + טענת יציאה: נתונים מוצפנים (bytes)
  + תיאור: מצפין נתונים באמצעות AES.
* decrypt:
  + טענת כניסה:
    - encrypted\_data (bytes): הנתונים המוצפנים לפענוח.
  + טענת יציאה: נתונים מפוענחים (bytes)
  + תיאור: מפענח נתונים מוצפנים באמצעות AES.

### **protocol.py**

Class: MessageParser

**תפקיד:** מחלקה זו אחראית על ניתוח ויצירת הודעות בהתאם לכללי הפרוטוקול המוגדרים. היא מגדירה את מבנה ההודעות, סוגי ההודעות ושיטות לקידוד ופענוח שלהן.

**תכונות:**

* PROTOCOL\_SEPARATOR (bytes): מפריד המשמש להפרדת שדות בהודעות הפרוטוקול.
* קבועים רבים (str או bytes): המגדירים סוגי הודעות שונים (למשל, CLIENT\_MSG\_SIG, MANAGER\_MSG\_EXIT וכו').

**פעולות:**

* encode\_str:
  + טענת כניסה:
    - msg (str או bytes): ההודעה לקידוד.
  + טענת יציאה: הודעה מקודדת (bytes)
  + תיאור: מקודד מחרוזת ל-bytes.
* protocol\_message\_construct:
  + טענת כניסה:
    - msg\_type (str): סוג ההודעה.
    - \*args: ארגומנטים נוספים להכללה בהודעה.
  + טענת יציאה: הודעה בנויה (bytes)
  + תיאור: בונה הודעה בהתאם לכללי הפרוטוקול.
* protocol\_message\_deconstruct:
  + טענת כניסה:
    - msg (bytes): זרם הבייטים של ההודעה.
    - part\_split (int, אופציונלי): מספר השדות להפרדה מתחילת ההודעה.
  + טענת יציאה: רשימה של שדות הודעה (רשימה של bytes)
  + תיאור: מפרק הודעה לשדותיה בהתאם לכללי הפרוטוקול.

Class: TCPsocket

**תפקיד:** מחלקה זו עוטפת את פונקציונליות ה-socket של TCP, ומספקת שיטות ליצירה, חיבור, שליחה, קבלה וסגירה של socket. היא מפשטת את פעולות ה-socket עבור המחלקות האחרות.

**תכונות:**

* MSG\_LEN\_LEN (int): אורך השדה המשמש לציון אורך ההודעה.
* \_\_sock (socket.socket): אובייקט ה-socket הבסיסי.
* \_\_ip (str, אופציונלי): כתובת ה-IP של ה-socket.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה:
    - sock (socket.socket, אופציונלי): אובייקט socket קיים.
  + טענת יציאה: אין
  + תיאור: יוצר socket TCP חדש או עוטף socket קיים.
* set\_timeout:
  + טענת כניסה:
    - time (int או float): זמן timeout בשניות.
  + טענת יציאה: אין
  + תיאור: מגדיר timeout ל-socket.
* get\_ip:
  + טענת כניסה: אין
  + טענת יציאה: כתובת ה-IP של ה-socket (str)
  + תיאור: מחזיר את כתובת ה-IP של ה-socket.
* create\_server\_socket:
  + טענת כניסה:
    - bind\_ip (str): כתובת ה-IP לקשירת השרת.
    - bind\_port (int): הפורט לקשירת השרת.
    - server\_listen (int): מספר החיבורים המקסימלי להאזנה.
  + טענת יציאה: אין
  + תיאור: מכין socket שרת TCP.
* server\_socket\_recv\_client:
  + טענת כניסה: אין
  + טענת יציאה: אובייקט socket של לקוח (socket.socket)
  + תיאור: השרת מקבל לקוח חדש.
* client\_socket\_connect\_server:
  + טענת כניסה:
    - dst\_ip (str): כתובת ה-IP של השרת.
    - dst\_port (int): הפורט של השרת.
  + טענת יציאה: אין
  + תיאור: מחבר socket לקוח לשרת.
* close:
  + טענת כניסה: אין
  + טענת יציאה: אין
  + תיאור: סוגר את ה-socket.
* send:
  + טענת כניסה:
    - data (bytes): הנתונים לשליחה.
  + טענת יציאה: אין
  + תיאור: שולח נתונים דרך ה-socket.
* recv:
  + טענת כניסה:
    - length (int): אורך הנתונים לקבלה.
  + טענת יציאה: נתונים שהתקבלו (bytes)
  + תיאור: מקבל נתונים מה-socket.

Class: client

**תפקיד:** מחלקה זו מרחיבה את TCPsocket כדי לטפל בתקשורת ספציפית בצד הלקוח. היא מוסיפה פונקציונליות להצפנה ופענוח של הודעות, וכן ניהול של הודעות "לא בטוחות".

**תכונות:**

* יורשת \_\_sock, \_\_ip מ-TCPsocket.
* \_\_encryption (EncryptionHandler, אופציונלי): מופע של EncryptionHandler לטיפול בהצפנה.
* \_\_unsafe\_msg\_cnt (int): מונה של הודעות "לא בטוחות" שהתקבלו.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה:
    - manager (bool, אופציונלי): דגל המציין אם זהו לקוח מנהל.
  + טענת יציאה: אין
  + תיאור: מאתחל אובייקט לקוח, כולל טיפול בהצפנה אם נדרש.
* set\_address:
* טענת כניסה:
  + - כתובת MAC (str) חדשה ללקוח
* טענת יציאה: אין
* תיאור: משנה את התכונה של האובייקט של הכתובת
* get\_address:
  + טענת כניסה: אין
  + טענת יציאה: מחזיר את הכתובת MAC
  + תיאור: מחזיר את הכתובת MAC של הלקוח/מנהל
* exchange\_keys:
  + טענת כניסה: אין
  + טענת יציאה: בוליאני המעיד האם ההחלפת מפתחות עברה בהצלחה
  + תיאור: מבצע החלפת מפתחות עם הצד השני המחובר
* connect:
  + טענת כניסה:
    - dst\_ip (str): כתובת ה-IP של השרת.
    - dst\_port (int): הפורט של השרת.
  + טענת יציאה: בוליאני המציין הצלחה או כישלון של החיבור.
  + תיאור: מתחבר לשרת ומטפל בחילופי מפתחות הצפנה.
* protocol\_recv:
  + טענת כניסה: אין
  + טענת יציאה: רשימה של נתוני הודעה (רשימה של bytes)
  + תיאור: מקבל הודעה מהשרת, מפענח אותה אם יש צורך ומחזיר את הנתונים.
* protocol\_send:
  + טענת כניסה:
    - msg\_type (str): סוג ההודעה.
    - \*args: נתוני ההודעה.
    - encrypt (bool, אופציונלי): דגל המציין אם להצפין את ההודעה.
  + טענת יציאה: אין
  + תיאור: שולח הודעה לשרת, מצפין אותה אם יש צורך.
* unsafe\_msg\_cnt\_inc:
  + טענת כניסה:
    - safety (int): סף הבטיחות למספר הודעות לא בטוחות.
  + טענת יציאה: בוליאני המציין אם לנתק את הלקוח.
  + תיאור: מגדיל את מונה ההודעות הלא בטוחות ומחזיר אם יש לנתק את הלקוח.
* reset\_unsafe\_msg\_cnt:
  + טענת כניסה: אין
  + טענת יציאה: אין
* תיאור: מאפס את מונה ההודעות הלא בטוחות

Class: Server

**תפקיד:** מחלקה זו מרחיבה את TCPsocket כדי לטפל בפעולות ספציפיות בצד השרת. היא מגדירה את כתובות ה-IP והיציאות של השרת ומספקת שיטות לקבלת חיבורים מלקוחות.

**תכונות:**

* יורשת \_\_sock, \_\_ip מ-TCPsocket.
* SERVER\_BIND\_IP (str): כתובת ה-IP שעליה השרת יופעל.
* SERVER\_BIND\_PORT (int): הפורט שעליו השרת יאזין.

**פעולות:**

* \_\_init\_\_:
  + טענת כניסה:
    - server\_listen (int, אופציונלי): מספר החיבורים המקסימלי להאזנה.
  + טענת יציאה: אין
  + תיאור: יוצר socket שרת TCP.
* recv\_client:
  + טענת כניסה: אין
  + טענת יציאה: אובייקט client
  + תיאור: מקבל לקוח חדש ומחזיר אובייקט client המייצג את החיבור.

קשיים:

Timeout- על סוקט קרנלי, היה קושי בלמצוא את הפקודה שמאפשרת לעשות זאת.

ביצוע פעולות blocking עם interrupt context היה תוקע את כל המערכת וכל המערכת מפסיקה לעבוד.

חיפוש פעולות שנקראות על ידי הקרנל.

פתרונות:

-בהתחלה הפיתרון היה להריץ תהליכון שבודק, לאחר מכן אחרי חיפוש מעמיק מצאתי פקודה קרנלית שנותנת לבצע timeout שליחת וקבלת הודעות.

-לאחר קריאת ספר מצאתי מבנה נתונים שפותח תהליכונים מאחרי הקלעים ומתזמן אותם ב process context לפי סדר – workqueue.

-הרבה פעולות של הקרנל שעליהם אני מבצע הוקים דורשות חיפוש ב source code, לכן הורדתי את ה source code על המחשב שלי וכל פעם שאני צריך למצוא פקודה אני מחפש שם.