

**מאת: יובל כוינה**

**ת.ז: 216825737**

**בית הספר: מקיף ג' אשדוד**

**שם המנחה: מורן טובול**

**שם הפרויקט: ExtremLink**

**שם החלופה: תכנון ותכנות מערכות**

**שנת ההגשה: 2025**



תוכן עניינים:

עמוד שער - 1

תוכן עניינים - 2

מבוא - 3

מבנה/ארכיטקטורה של הפרוייקט - 6

מחלקות הפרוייקט - 8

תרשים uml- 13

מדריך למשתמש - 15

בסיסי נתונים - 27

מדריך למפתח - 29

רפלקציה - 48

ביבליוגרפיה - 49

מבוא:

תיאור תכולת הספר:

ספר הפרויקט שלי כולל מבוא המציג את מטרות הפרויקט, תהליך המחקר והאתגרים המרכזיים שבהם נתקלתי במהלך העבודה. מעבר לכך, הספר כולל תיאור של ארכיטקטורת הפרויקט, מדריך למשתמש, פירוט על מסדי הנתונים שבניתי, וכן מדריך למפתחים הכולל הסברים מעמיקים על מרכיבי הפרויקט השונים. בסיום הספר מצורפים רפלקציה אישית וביבליוגרפיה.

רקע על הפרויקט:

הפרויקט שלי נקרא: ExtremLink, והוא מהווה פלטפורמה להשתלטות מרחוק על מחשבים אשר יצרתי. כלומר, תוכנה המבוססת על בסיס מודל שרת-לקוח המתוכנת בשפת #C בסביבת העבודה WPF - Windows Presentation Foundation. בנוסף, השרת כולל גם מאגרי מידע מסוג SQL, כאשר מטרתם הוא לאחסן את נתוני המשתמשים של הלקוחות ואת ההקלטות של ההשתלטויות של אותם המשתמשים.

הפרויקט מאפשר תקשורת בין המשתמשים אשר הוזנו במערכת, כאשר התקשורת נעשית על ידי שיתוף מסך המחשב של הלקוח המשתף אל הלקוח השולט. לשם הרשמה למערכת ויצירת חשבון חדש, על הלקוח להזין שם משתמש, סיסמה, את שמו הפרטי, את שם המשפחה שלו, את מקום מגוריו, מספר הטלפון שלו ואת כתובת המייל שלו. לאחר שנתוני הלקוח הוזנו, הם מתאכסנים בתוך מאגר המידע של הלקוחות שנמצא בצד של השרת ובאפשרותם להזדהות במערכת באמצעות שם המשתמש שלהם והסיסמה בלבד. במידה והלקוח שכח את הסיסמה למשתמש שלו באפשרותו לשחזרה באמצעות בקשה לשליחה חוזרת של הסיסמה לכתובת המייל שלו ובכך לשחזר את סיסמתו. לאחר ש 2 לקוחות שונים (משתף ושולט) מתחברים למערכת באותה רשת פנימית, באפשרותם להתחיל את השליטה מרחוק, לעצור אותה באופן זמני, לעצור אותה באופן מוחלט, להמשיך אותה ואף להקליטה במידת הצורך! לאחר מכן במידה ואחד הלקוחות רוצה לצפות בהקלטה שביצע (בתור תוקף) באפשרותו לבחור באחת מבין ההקלטות המשויכות לו ממאגר המידע של ההקלטות ולצפות בה בנגן הוידאו בתוכנה עצמה.

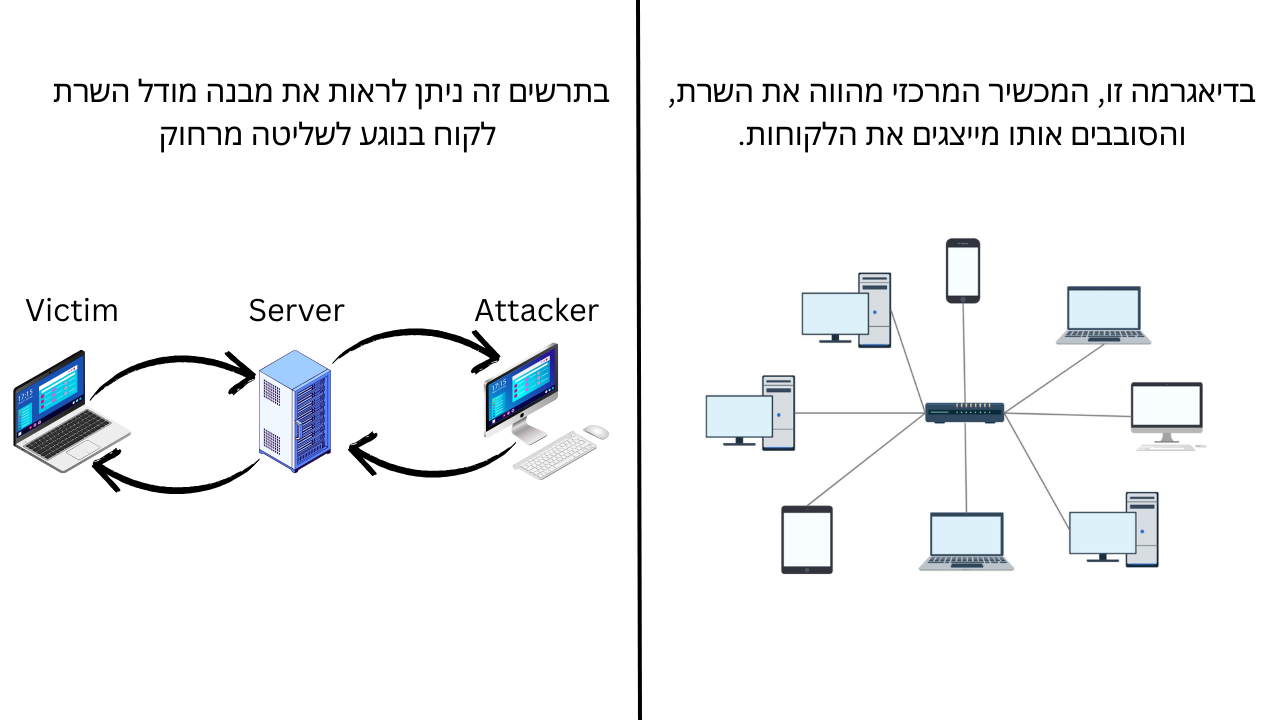
תהליך המחקר:

המחקר שביצעתי במסגרת הפרויקט התמקד בלמידה מעמיקה של מודל השרת-לקוח. מודל זה מהווה ארכיטקטורת תוכנה בעולם המחשוב המבוזר, המגדירה את אופן הפעולה והקשרים בין תוכנות שונות הפועלות יחד דרך רשת דיגיטלית. מודל זה מבוסס על חלוקת משימות בין שני רכיבים עיקריים: השרת והלקוח.

השרת הוא רכיב תוכנתי פסיבי, אשר מאזין לרשת באופן קבוע וממתין לבקשות. תפקידו הוא לספק שירותים או משאבים לבקשות שמתקבלות מהלקוחות – שהם החלק הפעיל במודל, הפונים לשרת לפי הצורך. לרוב, הלקוח מהווה את ממשק המשתמש, ומופעל על ידי המשתמש עצמו.

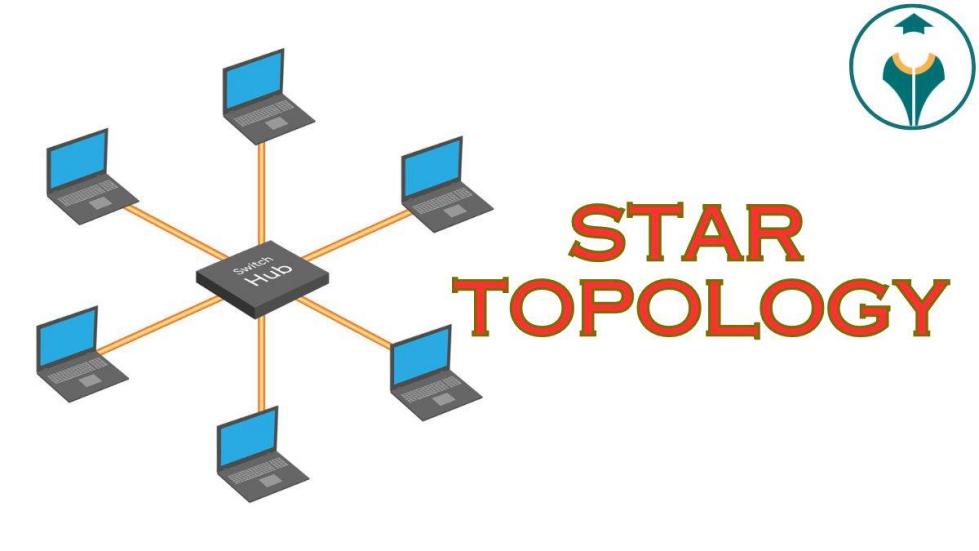
התקשורת בין השרת ללקוח נעשית באמצעות העברת הודעות בהתאם לפרוטוקול תקשורת מוגדר מראש. כאשר הלקוח מבקש שירות, הוא שולח בקשת התחברות אל השרת. השרת, אשר מאזין לבקשות אלה, מקבל את הפנייה ומשיב בהתאם. במבנה זה, הלקוח מוגדר כיוזם התקשורת, ואילו השרת הוא זה שממתין לבקשות ומגיב להן. היותו של מודל שרת לקוח ארכיטקטורה מבוזרת, מהווה יתרון חשוב ביותר, כאשר הוא יכול לנצל ביעילות מספר רב של משימות. בנוסף, ניתן לשלב שרת או לקוח חדשים לתוך המערכת ולשדרג קיימים, מבלי להשפיע על חלקים אחרים במערכת ובצורה קלה ביותר.

מודל השרת-לקוח יכול לפעול בשתי תצורות עיקריות. לרוב, תוכנות השרת והלקוח פועלות על גבי מחשבים נפרדים, והתקשורת ביניהן מתבצעת דרך רשת מחשבים. עם זאת, קיימים גם מקרים בהם שתי התוכנות פועלות על אותו מחשב מקומי, כאשר התקשורת מתבצעת באופן פנימי.

מודל זה הפך לאבן יסוד בתחום רשתות המחשבים, ורבים מהיישומים המודרניים מבוססים עליו. דוגמאות לכך כוללות שירותי גלישה באינטרנט, גישה למסדי נתונים, וכמובן – גם הפרויקט שפיתחתי.

הפרויקט שלי מתבסס על מודל השרת-לקוח, ומשמש כפלטפורמה לתקשורת בין מחשבים. במסגרת פעילותו, השרת אחראי על קבלת נתונים מהלקוחות ושליחתם ללקוחות אחרים. לדוגמה, בעת השתלטות מרחוק, הלקוח המשתף שולח לשרת רצף פריימים ממסך מחשבו, והשרת מעביר את המידע הזה אל מחשב הלקוח השולט (התוקף).

טופולוגית רשת מתארת את הסידור הממשי או הווירטואלי של הרכיבים השונים ברשת מחשבים. דוגמא לטופולוגית רשת הינה טופולוגית כוכב, המכונה גם **רשת כוכבים**, והיא אחת מהתקנות הרשת הנפוצות ביותר. בתצורה זו, כל **הצמתים (nodes)** מתחברים לרכיב תקשורת אחד הנמצא במרכז באמצעות כבלים, כאשר רכיב התקשורת המרכזי יכול להיות: מרכזת, מתג, נתב, מחשב וכדומה. רכיב תקשורת זה מהווה ופועל כשרת, כאשר המכשירים הסובבים אותו פועלים כלקוחות. נוצרת כאן תצורה בצורת כוכב, וכך טופולוגית רשת זו קיבלה את שמה.

**לטופולוגית כוכב** מספר יתרונות בולטים. ראשית, היא נחשבת לקלה יחסית לתחזוקה שוטפת, מה שמקל על ניהול הרשת לאורך זמן. בנוסף, ניתן להוסיף או להסיר ממנה מכשירים בצורה פשוטה וללא צורך בהתקנות מסובכות. יתרון חשוב נוסף הוא שעצירה בפעולתו של אחד מהמכשירים ההיקפיים אינה משפיעה על שאר המכשירים ברשת, כך שהרשת כולה ממשיכה לתפקד באופן תקין.

למרות יתרונותיה, **לטופולוגית כוכב** יש גם כמה חסרונות. ראשית, היא דורשת שימוש רב יותר בכבלים לעומת טופולוגיות אחרות, מה שעלול להוביל לבזבוז. בנוסף, המרכזיות של הצומת המרכזי הופכת אותו לנקודת תורפה: תקלה בו תשבית את כל הרשת. מעבר לכך, יכולות הרשת תלויות באופן ישיר בביצועי המכשיר המרכזי, אשר מגביל גם את מספר הצמתים שניתן לחבר לרשת.

אתגרים מרכזיים:

הבעיה שאיתה התמודדתי ופתרונותיה -

הייתי צריך להתמודד עם מספר אתגרים במהלך עבודתי על הפרויקט, כאשר הבעיה העיקרית שהייתי צריך להתמודד עימה הייתה העברת הפריימים של השידור החי של שיתוף המסך בפרויקט בין המחשבים. כדי להתמודד עם בעיה זו, היה עלי לבצע תהליך חקירה לעומק על הנושא. החקירה כללה חיפוש על תקשורת נתונים, פרוטוקולי תקשורת המאפשרים לבצעה העברה מהירה של נתונים גדולים ושיטות להמרת הנתונים. מושגים אלו המגיעים מעולמות מדעי המחשב והתקשורת מתעסקים בתקשורת בין מחשבים הכוללת העברת נתונים גרפיים כגון תמונות וודיאו בזמן אמת תוך מימוש מספר עקרונות המאפשרים ייעול מירבי של כלל התהליכים המעורבים בתקשורת הזאת. בפרוייקט שלי העוסק בהעברת קבצי תמונות וודיאו בזמן אמת היה עליי לערוך מחקר בנוגע לפרוטוקולי התקשורת המותאמים ביותר למשימה מה שהוביל אותי בסופו של דבר להשתמש בפרוטוקול UDP. פרוטוקול זה שנמצא בשכבת התעבורה (כחלק ממודל השכבות שעליו מבוססת התקשורת האינטרנטית) נמצא כיעיל ביותר להעברת נתונים רחבי היקף בזמן קצר על חשבון אמינות המידע שהתקבל (בחלקו או בשלמותו) מכיוון שהוא אינו בודק את אמינות המידע על מנת לצמצם זמן חשוב. לכן השימוש המושכל בפרוטוקול היה אבן דרך משמעותית בפתרון הבעיה שאיתה התמודדתי. בנוסף לכך, בנוגע להמרת הנתונים היה עליי לחקור על שיטות המרה של נתונים בפורמט וויזואלי לידי פורמט בינארי (שפה המבוססת על בסיס 2 המכיל את הספרות 0 ו 1) הידוע ככזה המשמש לתקשורת בין מחשבים. לכן הבנתי שהפתרון הטוב ביותר שאיפשר לי המרה של תמונות ווידאו לפורמט בינארי הוא יצירה של קבצים זמניים שיאחסנו באופן זמני את האמצעים הוויזואלים ולאחר מכן שליחה של התוכן הבינארי של הקבצים עצמם. ובכך פתרתי את הבעיה המרכזית שלי בפרויקט!

הסיבות לבחירת הפרויקט ומוטיבציה לעבודה -

אני בחרתי בפרויקט ExtremLink, מכיוון שהיה לי רצון וכמיהה ליצור פרויקט המיישם עקרונות מעולמות תקשורת הנתונים והסייבר שאני מאוד מתחבר אליהם ואף חוקר עליהם בשעות הפנאי. בנוסף לכך, גם ברצוני היה ליצור פרויקט גדול ומסודר שאוכל להוסיפו לתיק הפרויקטים שלי ולהשתמש בו בתור אסמכתא לשימוש עתידי בתעשיית היי-טק.

מה גם, הפרויקט איפשר לי לצבור ניסיון מעמיק ורב בתכנות ככלל ובשפת סי-שארפ בפרט. ניסיון זה כמובן יקנה לי יתרון בשוק העבודה וגם ייאפשר לי ליצור פרויקטים נוספים בתחום ואף להגיע לתוצרים שלא חלמתי שאוכל ליצור!

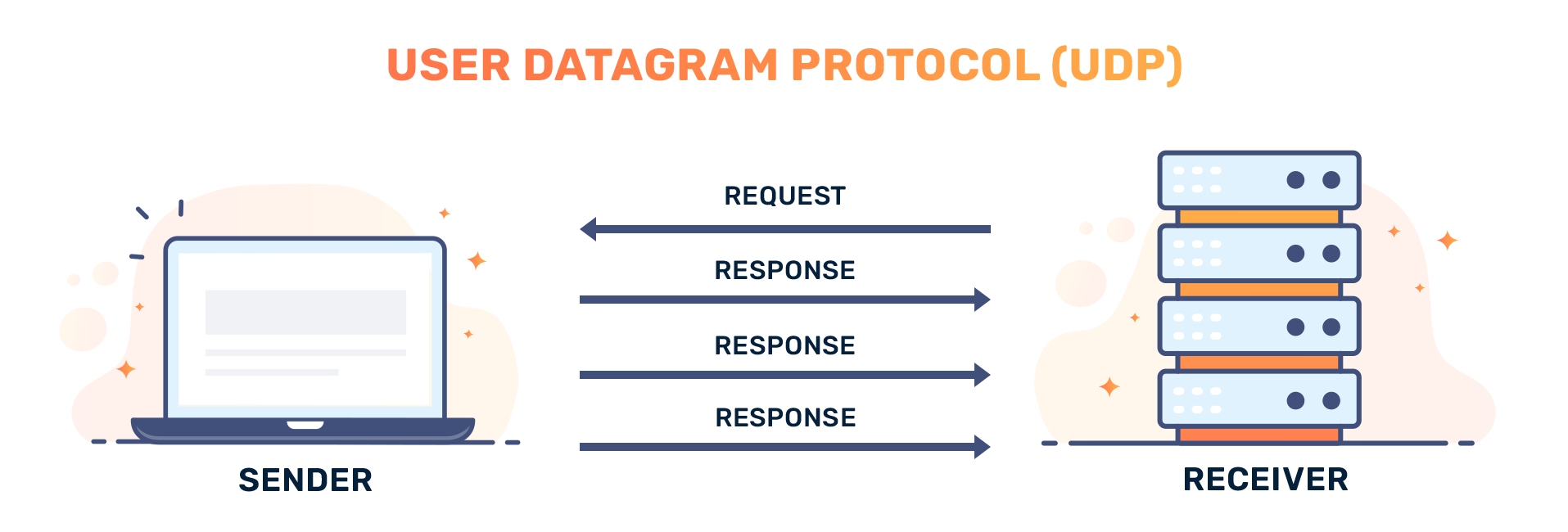
הצורך שעליו עונה הפרויקט -

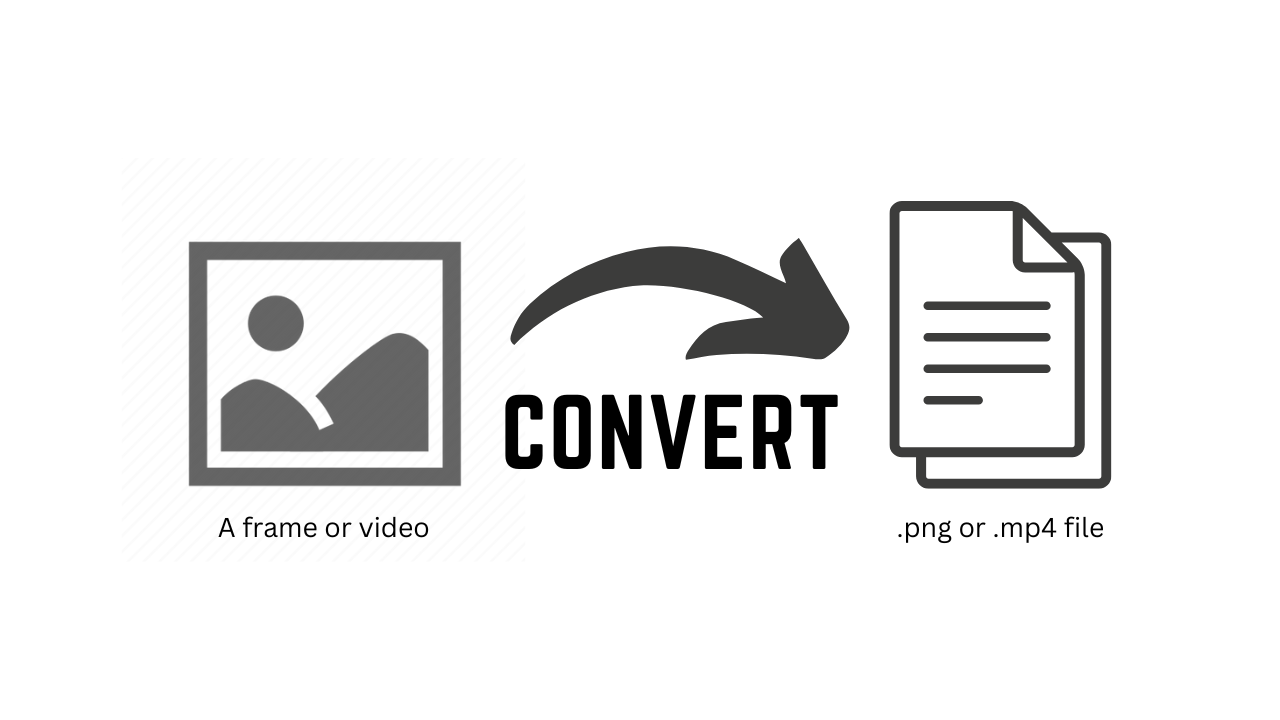
הפרויקט שלי עונה על הצורך לאמצעי תקשורת זמין ומהיר, כאשר לקוח מסוים נתקל בבעיה כלשהי במחשבו האישי וברצונו שמפעיל מרחוק כגון טכנאי או כמגיש שירות לקוחות יתקן את התקלה מרחוק ללא צורך בהגעה פיזית. פתרון זה יפתח צוהר לשירותים חדשים בעולמות טכנאות המחשבים ויקל על הלקוחות.

מבנה / ארכיטקטורה של הפרויקט:

הצגת הפתרון המוצע והסיבות לבחירתו -

הביעה שאיתה התמודדתי הייתה העברת נתונים וויזואלים בין מחשבים, לצורך יצירת האפשרות לשיתוף מסך בין לקוחות. הדרך שאיתה התמודדתי הייתה חיפוש מעמיק ברחבי האינטרנט, באתרים העוסקים במדעי המחשב, בפורמים ואף לשאול מומחים העוסקים בתחום. הפתרונות כאמור מבוססים על מודל שרת-לקוח הם **שימוש בפרוטוקול UDP** ו**המרת נתונים לקבצים ושליחתם.**





מחלקות הפרויקט

צד שרת -

|  |  |  |
| --- | --- | --- |
| **קלט ופלט** | **תפקיד המחלקה** | **שם המחלקה** |
| **המחלקה לא מקבלת כלום והפלטים שלה כוללים מערכי בתים (byte arrays) המייצגים פריימים של וידאו או קבצי סשן מוקלטים, מחרוזות בפורמט JSON המכילות מידע על סשנים, רשימות של אובייקטים מסוג סשן, ערכים לוגיים (true/false) המצביעים על הצלחה או כישלון בפעולות מול מסד הנתונים, אובייקטים של חיבור למסד נתונים, וכן פלטים שאינם מוחשיים כמו שליחת נתונים דרך רשת, עדכון מידע אצל הלקוחות, או רישום פעולות ביומן לוגים – מבלי להחזיר ערך ישיר.** | **מחלקת Server אחראית על ניהול התקשורת בין לקוחות השולט והנשלט בתכונה. היא מטפלת בהתחברות ויצירת הלקוחות, בהעברת פקודות וידאו, מקלדת ועכבר, ובניהול משתמשים והקלטות מול מסד הנתונים, תוך התבססות על תבנית העיצוב Singleton.** | **Server** |
| **המחלקה מקבלת בעת יצירתה מחרוזת שמייצגת כתובת IP של שרת, וסוג של לקוח (תוקף או קורבן). הפלטים שלה כוללים הודעות בפורמט מפורק כמבני נתונים מתוך תקשורת TCP ו־UDP, אובייקטים של שקעי רשת (Sockets) שמייצגים ערוצי תקשורת שונים, כתובות IP ומידע מזהה של הלקוח. בנוסף, מתקיימות פעולות שלא מחזירות ערך ישיר אך גורמות לחיבור לרשת, שליחת הודעות, קבלת נתונים מהלקוח, וקביעת נקודת הקצה של תקשורת UDP בהתאם לסוג הלקוח.** | **מחלקת Client מייצגת לקוח (שולט או נשלט) בצד השרת. היא אחראית על ניהול החיבורים מסוג TCP ו-UDP מול השרת, קבלת ושליחת הודעות, שמירת פרטי המשתמש וה-IP של הלקוח, ותמיכה בתקשורת עבור זרמי וידאו וסשנים מוקלטים. המחלקה מהווה את נקודת הקצה בתקשורת מצד השרת מול הלקוח הרלוונטי.** | **Client** |
| **המחלקה מקבלת בעת יצירתה תאריך הקלטה, שם משתמש ולעיתים גם מזהה ייחודי של סשן. הפלטים שלה כוללים מערכי בתים (byte arrays) שמייצגים תוכן וידאו מוקלט, מחרוזות עם שמות משתמשים, מזהים מספריים ותאריכים. בנוסף, מתבצעות פעולות שלא מחזירות ערך ישיר אך מבצעות שמירה של נתוני הסשן למסד נתונים, פתיחת חיבורים למסד וכתיבת הודעות ליומן לוגים לצורך תיעוד הפעולות.** | **מחלקת Session מייצגת סשן של שליטה מרחוק הכולל הקלטת וידאו של פעילות המשתמש. היא שומרת את תוכן הווידאו כ־byte array, את זמן ההקלטה, את שם המשתמש ואת מזהה הסשן (ID). בנוסף, היא מאפשרת להעלות את הסשן למסד הנתונים לצורכי אחסון ושימוש מאוחר יותר.** | **Session** |
| **המחלקה אינה מקבלת כלום בעת יצירתה והפלט היחיד שלה הוא שם המשתמש בפורמט string.** | **מחלקת User מייצגת משתמש של לקוח במערכת. היא שומרת את שם המשתמש ומספקת ממשק פשוט לגישה למידע זה. המחלקה משמשת כתשתית בסיסית לזיהוי וניהול משתמשים בצד השרת.** | **User** |
| **המחלקה לא מקבלת קלט בעת יצירתה, והפלטים שלה כוללים מחרוזות המייצגות הודעות לוג, וכן פעולות שאינן מחזירות ערך ישיר אך גורמות לעדכון פנימי של יומן הרישום. בנוסף, מתבצעת גישה למופע יחיד של המחלקה המנוהל באופן עצמאי, תוך שמירה על עקביות של המידע לאורך חיי היישום.** | **המחלקה Log מנהלת יומן רישום (Log) של הודעות בצד השרת, באמצעות שמירה של רשימת הודעות תוך שימוש בתבנית העיצוב Singleton, המחלקה מאפשרת הוספה של הודעות ליומן ואיחסונם.** | **Log** |

צד לקוח -

|  |  |  |
| --- | --- | --- |
| **קלט ופלט** | **תפקיד המחלקה** | **שם המחלקה** |
| **הפונקציות במחלקה Client פועלות על הנתונים הפנימיים שלה, כגון חיבורי הרשת וכתובות השרת, ללא צורך בקלט חיצוני. הפלטים המוחזרים משתנים בהתאם להקשר הפעולה - החל מנתונים בינאריים של פריימי וידאו וקבצי סשן, דרך מידע מובנה בפורמט JSON על מצב החיבורים, ועד רשימות אובייקטים המייצגות סשנים פעילים. הפונקציות מחזירות ערכים בוליאניים המעידים על הצלחה או כישלון בביצוע פעולות קריטיות כמו עדכון מסד נתונים, או לחילופין לבצע פעולות רקע כגון העברת נתונים, סנכרון בין לקוחות ורישום אירועים במערכת הלוגים - כל זאת ללא החזרת ערך ישיר.** | מחלקת **Client** מייצגת לקוח שמתחבר לשרת מרחוק. היא אחראית על יצירת והגדרת החיבורים בין המחשב של המשתמש לשרת, כולל חיבורים רגילים וחיבורים מיוחדים לפעולות ספציפיות. המחלקה מאפשרת קבלת פרטים מהשרת כמו כתובתו והודעות וכמו כן אחראית לשלוח אליו הודעות ספציפיות לשרת וללקוחות אחרים דרך השרת. | Client |
| **המחלקה Attacker לא מקבלת כלום, והפלטים שלה כוללים מחרוזות המייצגות כתובת IP של הקורבן, תמונות מסוג BitmapImage המייצגות פריימים, מערכי בתים (byte arrays) המייצגים פריימים של וידאו או תכני סשנים, רשימות של אובייקטים עם פרטי הודעות שהתקבלו, פעולות שאינן מחזירות ערכים ישירים כגון שליחת הודעות ללקוח דרך סוקטים (TCP/UDP), קבלת הודעות מהלקוח, שמירת פריימים כקבצי PNG, יצירת קובצי MP4 זמניים מסשנים מוקלטים, עיבוד קלטי עכבר ומקלדת מהשרת, ניהול מידע על סשנים עבור המשתמש, ועדכון פרטי כתובת ה-IP של הקורבן על פי מידע שמגיע מהשרת.** | מחלקת **Attacker** מייצגת תוקף המחובר לשרת מרחוק ומנהל תקשורת עם קורבן. היא יורשת ממחלקת Client ומתמקדת בשליטה ובקרה על הקורבן - כולל קבלת תמונות מסך בזמן אמת, ניהול הקלטות, וטיפול בפקודות מקלדת ועכבר. המחלקה משתמשת בתבנית העיצוב **Singleton** ומטפלת בשליחה וקבלה של הודעות דרך חיבורי רשת ייעודיים, תוך פיצול הנתונים לחבילות והרכבתם מחדש. | Attacker |
| **המחלקה Victimלא מקבלת כלום, והפלטים שלה כוללים מחרוזות המייצגות כתובת IP של התוקף, מערכי בתים (byte arrays) המייצגים פריימים של וידאו או קבצי סשן מוקלטים, רשימות של אובייקטים עם פרטי הודעות שהתקבלו, תמונות הנשלחות כתמונות PNG מקובץ או ממסך משותף, פעולות שאינן מחזירות ערך ישיר כמו שליחת הודעות או פריימים דרך סוקטים (TCP/UDP), קבלת פקודות מהשרת שמעדכנות את מצבי העכבר והמקלדת, קבלת פרטי סשנים ועדכונם עבור המשתמש, יצירת קובצי PNG מהמסך, קבלת תוכן קבצים, ריקון מחרוזת תגובת השרת, וכן פעולות של שמירת פריימים זמניים ושליחתן.** | מחלקת **Victim** מייצגת קורבן המחובר לשרת מרחוק ומאפשר שליטה בו על ידי תוקף. היא יורשת ממחלקת Client ומתמקדת בטיפול בפקודות מהתוקף - כולל שליחת תמונות מסך, הקלטות, וקבלת פקודות מקלדת ועכבר. המחלקה משתמשת בתבנית העיצוב **Singleton** ומטפלת בחלוקת הנתונים לחבילות לפני שליחתם דרך הרשת. בנוסף, היא מספקת יכולת תגובה לשרת עם סטטוס פעולות כמו אישור קבלת פקודות או עדכון מצב ההקלטה. | Victim |
| **המחלקה Userלא מקבלת כלום והפלטים שלה כוללים מחרוזות המייצגות את שם המשתמש, טיפוסי אובייקט שמייצגים את סוג הלקוח (TypeOfClient) ורשימות של אובייקטים מסוג סשן (Session) המייצגות את הסשנים של המשתמש.** | מחלקת **User** מייצגת את המשתמש במערכת, בין אם הוא תוקף או קורבן. היא כוללת את שם המשתמש, סוג הלקוח (תוקף/קורבן) ורשימת ההקלטות שלו. המחלקה מאפשרת גישה ועדכון לפרטי המשתמש בצורה מבוקרת. | User |
| **המחלקה Session מקבלת אובייקט שמייצג תאריך מסוג DateTime ומספר מזהה מסוג int. הפלטים שלה כוללים מחרוזות בפורמט JSON לבקשת מידע מהשרת על סשנים, שליחה של הודעות לשרת דרך חיבור TCP מבלי להחזיר ערך ישיר, יצירה של קובץ וידאו זמני מסוג MP4 מתוכן של מערך בתים (byte array), וכן רשימות של אובייקטים מסוג Session המייצגות תכונות של סשנים על פי מחרוזת JSON שהתקבלה.** | מחלקת Session מייצגת הקלטת מסך (session) במערכת. היא כוללת את תאריך ההקלטה, תוכן הווידאו (כמערך בתים) ומזהה ייחודי. המחלקה מספקת פונקציונליות לשליחת בקשות לשרת - הן לקבלת רשימת ההקלטות של משתמש מסוים והן לקבלת תוכן הקלטה ספציפית לפי ID. | Session |
| **המחלקה AttackerSession מקבלת שני מספרים שלמים המייצגים את רוחב וגובה הווידאו. הפלטים שלה כוללים קבצי וידאו זמניים בפורמט MP4 המאוחסנים כקבצים פיזיים ודחוסים למערכי בתים (byte arrays), שליחה של תוכן הווידאו לשרת באמצעות חיבור TCP במנות מפוצלות לפי גודל חבילה, והפעלת או עצירת תהליך הקלטת הווידאו מבלי להחזיר ערך ישיר.** | מחלקת **AttackerSession** אחראית על הקלטת וניהול סרטוני המסך מהצד התוקף. היא מספקת יכולת להתחיל ולהפסיק הקלטות, להוסיף פריימים לסרטון בזמן אמת, ולשלוח את ההקלטות לשרת. | AttackerSession |
| **המחלקה CustomMouseAttacker לא מקבלת קלט והפלטים שלה כוללים מחרוזות בפורמט JSON שמייצגות פקודות עכבר כמו תזוזה, לחיצה שמאלית או לחיצה ימנית ושליחת הפקודות הללו לשרת דרך חיבור TCP מבלי להחזיר ערך ישיר.** | מחלקת **CustomMouseAttacker** מייצגת את ממשק העכבר של התוקף ומאפשרת שליטה מרחוק בעכבר הקורבן. היא בנויה לפי תבנית העיצוב **Singleton** ומנהלת את מיקום הסמן ופעולות העכבר. המחלקה יוצרת פקודות JSON הכוללות את סוג הפעולה (תזוזה/לחיצה) ומיקום הסמן, ושולחת אותן דרך השרת לקורבן. | CustomMouseAttacker |
| **המחלקה CustomMouseVictim מקבלת בעת יצירתה ערכים התחלתיים של נקודת עכבר (Point) ופקודת עכבר (VictimMouseCommands) שמוגדרים כברירת מחדל לאפס ו-CommandLess בהתאמה. הפלטים שלה כוללים עדכון מיקום העכבר לפי נקודת ציון חדשה, תזוזת העכבר במערכת ההפעלה למיקום מוגדר, סימולציה של לחיצת עכבר שמאלית או ימנית, או ביצוע פעולה על פי פקודת עכבר קיימת — כל זאת ללא החזרת ערך ישיר אלא דרך פעולות על המערכת באמצעות WIN API.** | מחלקת **CustomMouseVictim** אחראית על ביצוע פעולות העכבר בפועל במחשב הקורבן. היא מקבלת פקודות מהתוקף (כמו תזוזה או לחיצות) ומתרגמת אותן לפעולות ממשיות על מערכת ההפעלה של הקורבן באמצעות קריאות ל **WIN API.** | CustomMouseVictim |
| **המחלקה CustomKeyboardAttacker מקבלת בעת יצירתה ערכים התחלתיים של מקש לחוץ (Key) ופקודת מקלדת (AttackerKeyboardCommands) שמוגדרים כברירת מחדל ל-None ו-CommandLess בהתאמה. הפלטים שלה כוללים יצירת מחרוזת בפורמט JSON שמייצגת לחיצת מקש, שליחת פקודות ללקוח דרך הודעות TCP, יידוע על לחיצות ושחרור מקשים באמצעות אירועים, והגדרת האזנה ללחיצות מקשים דרך API של Windows – מבלי להחזיר ערך ישיר בפונקציות אלו.** | מחלקת **CustomKeyboardAttacker** מייצגת את ממשק המקלדת בצד התוקף ומאפשרת שליטה מרחוק במקלדת הקורבן. המחלקה בנויה על פי תבנית העיצוב **Singleton.** | CustomKeyboardAttacker |
| **המחלקהCustomKeyboardVictim מקבלת בעת יצירתה ערכים התחלתיים של מקש לחוץ (Key) ופקודת מקלדת (VictimKeyboardCommands) שמוגדרים כברירת מחדל ל-None ו-CommandLess בהתאמה. הפלטים שלה כוללים סימולציה של לחיצת ושחרור מקש במערכת ההפעלה באמצעות קריאות ל-API של Windows, וביצוע פקודת מקלדת נוכחית אם קיימת — כל זאת מבלי להחזיר ערך ישיר מהפונקציות.** | מחלקת **CustomKeyboardVictim** מייצגת את ממשק המקלדת בצד הקורבן ומבצעת בפועל את פקודות המקלדת שמתקבלות מהתוקף. המחלקה בנויה על פי תבנית העיצוב **Singleton.** | CustomKeyboardVictim |
| **המחלקה SoundManager לא מקבלת כלום ולא מחזירה כלום.** | מחלקת **SoundManager** אחראית על ניהול ושליטה בקבצי האודיו בתוכנה. היא בנויה על פי תבנית העיצוב **Singleton** ומספקת ממשק פשוט להפעלת מוזיקה בתוכנה. | SoundManager |

מבנה נתונים בהם נעשה שימוש -

במהלך הפרויקט, נעשה שימוש במבני הנתונים: מערך ורשימה, כאשר השימוש במערך היה כדי ליצור מערך של ביתיים שייצג תוכן של קבצים מסוג תמונות (פריימים) והקלטות (סשנים) שמשומשות בפרוייקט עת העברת הפריימים בין מהחשבים והעברת האחסון והצפייה בהקלטות.

ארכיטקטורת רשת -

**תיאור פרוטוקולי התקשורת של מודל שרת לקוח –**

רשת האינטרנט מהווה תשתית עולמית עצומה, המכילה אינספור אתרי אינטרנט הנגישים לכלל הציבור. לכל אתר קיים שם דומיין (**Domain Name**), לדוגמה: facebook.com, שהוא שם ייחודי המאפשר לזהות ולהבדיל בין אתרים שונים באינטרנט. מאחורי כל שם דומיין נמצאת כתובת IP שהיא למעשה הכתובת האמיתית של השרת עליו מאוחסן האתר (הדומיין קיים לצורכי נוחות בגישה ישירה לשרתים אלו).

מכיוון שמחשבים מתקשרים באמצעות מספרים, אך בני אדם זקוקים לשמות מילוליים שקל לזכור ולהקליד, נדרש מנגנון תרגום בין השניים. כאן נכנס לפעולה פרוטוקול ה־**DNS (Domain Name System),** אשר מאפשר המרה של שמות דומיין לכתובות IP. כך, ניתן לגלוש לאתרים בקלות על ידי הקלדת שם דומיין בלבד.

פרוטוקול ה־DNS מתבסס על מערכת שרתים מבוזרת, כאשר כל שרת **DNS** אחראי לתחום (**Zone**) מסוים. תחום זה מהווה טבלה הכוללת סוגי רשומות עם שדות רלוונטיים (כמו כתובות של שרתי דואר). לצורך סנכרון בין שרת DNS ראשי לשרת משני, מתבצעת העברת תחומים (Zone Transfer) באמצעות פרוטוקול **TCP** על פורט **53**.

בנוסף, מודל השרת-לקוח מתבסס על חבילת פרוטוקולי התקשורת **TCP/IP**.

**TCP (Transmission Control Protocol)** הוא פרוטוקול אמין שמוודא שכל הנתונים אכן מגיעים ליעד בשלמותם. הוא מחלק את המידע לחבילות (packets), מוסיף להן מספרים סידוריים, ומוודא שכל החבילות הגיעו ונאספו מחדש בסדר הנכון. אם חסרה חבילה, TCP מבקש אותה שוב. לכן, הוא מתאים ליישומים שדורשים אמינות גבוהה כמו דפדפני אינטרנט או העברת קבצים.

**IP (Internet Protocol)** אחראי על ניתוב החבילות ליעדן. הוא אינו מבצע בדיקות אמינות, אלא רק מטפל בכתובת היעד ומוודא שהחבילה מועברת הלאה לכיוון הנכון. פרוטוקול זה יעיל ומהיר מאוד, ומהווה את הבסיס לתעבורת הרשת כולה.

**TCP/IP** יחד מהווים את הבסיס לאינטרנט, ומודל השרת-לקוח משתמש בהם לתקשורת בין מחשבים – כך שהלקוח שולח בקשה לשרת ומקבל תגובה.

אחד היתרונות במודל זה הוא שכל בקשה מהלקוח נחשבת כפעולה נפרדת, שאינה תלויה בבקשה קודמת. כתוצאה מכך, התקשורת בין מחשבים מתאפשרת בצורה דינאמית, כאשר משאבי הרשת מנוצלים בצורה יעילה.

כמו כן, המודל יכול להשתמש בפרוטוקולים נוספים לפי מודל **ה־OSI**. לדוגמה:

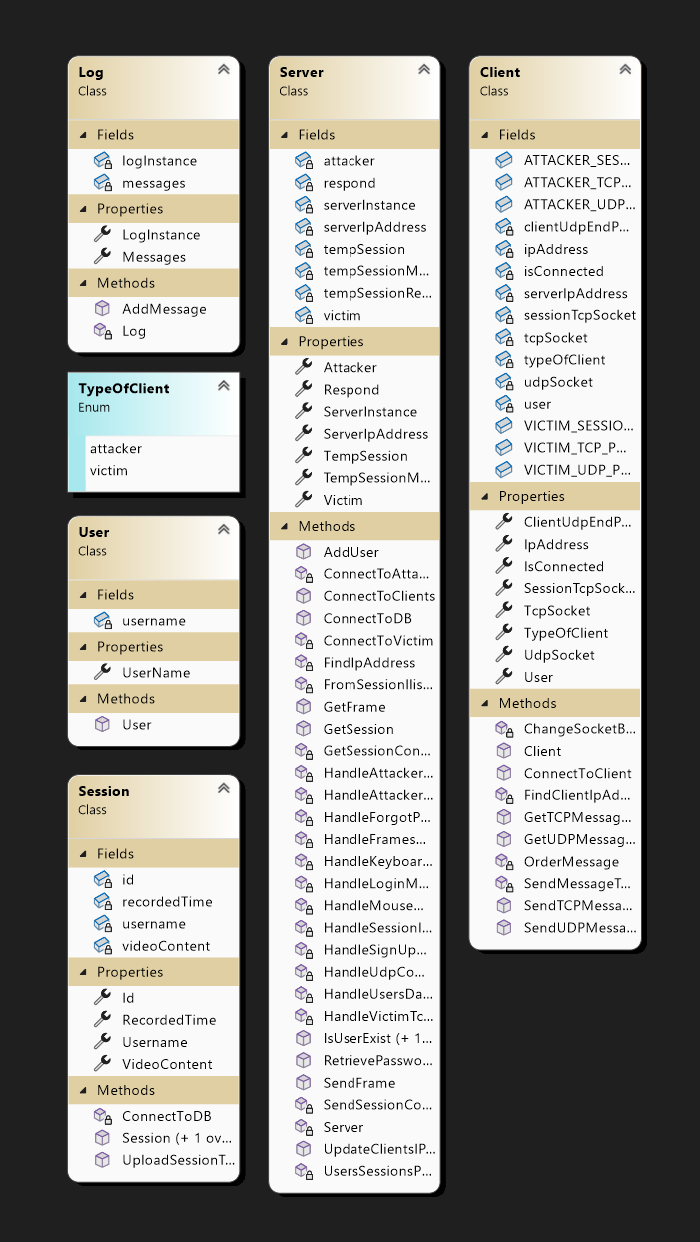
**UDP (User Datagram Protocol)** – פרוטוקול נוסף משכבת התעבורה, אשר גם בו נעשה שימוש בפרויקט שלי. UDP שונה מ־TCP בכך שהוא אינו בודק את שלמות המידע שמתקבל – ולכן הוא מהיר יותר, אך גם פחות אמין. אם במהלך התקשורת אובדות חבילות מידע, UDP לא מנסה לשלוח אותן שוב, אלא ממשיך הלאה. הדבר הופך אותו לאידיאלי עבור יישומים שבהם מהירות חשובה יותר מהשלמות – כמו הזרמת וידאו בזמן אמת, או שליחת פריימים בפרויקט שלי.

**הטכנולוגיות שבהן נעשה שימוש בפרויקט:**

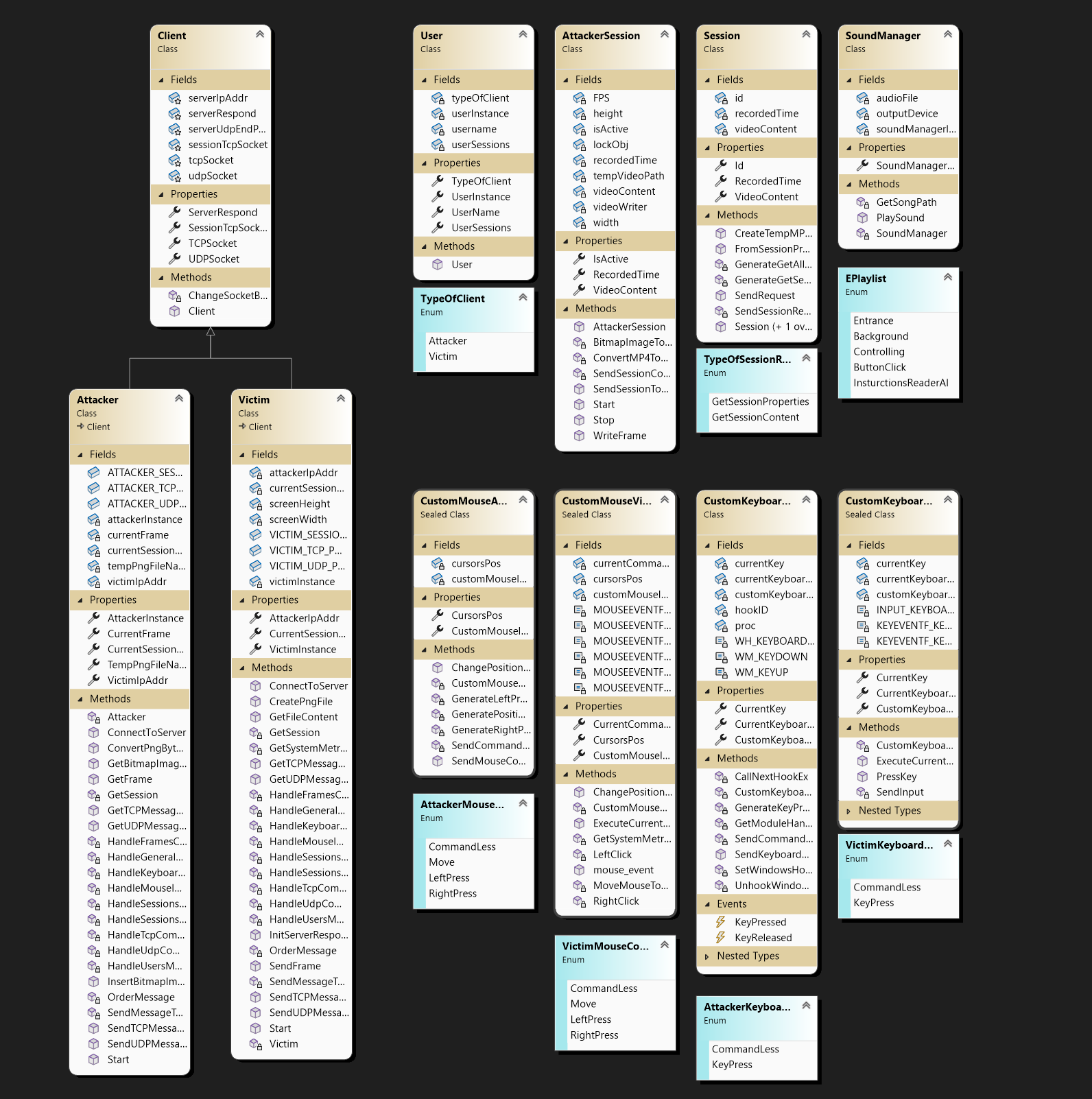
הטכנולוגיות שבהן השתמשתי בעת יצירת הפרויקט הינן: שפת התכנות C#, פלטפורמת הפיתוח WPF וטכנולוגית אחסון הנתונים SQL.

**תרשים UML-**

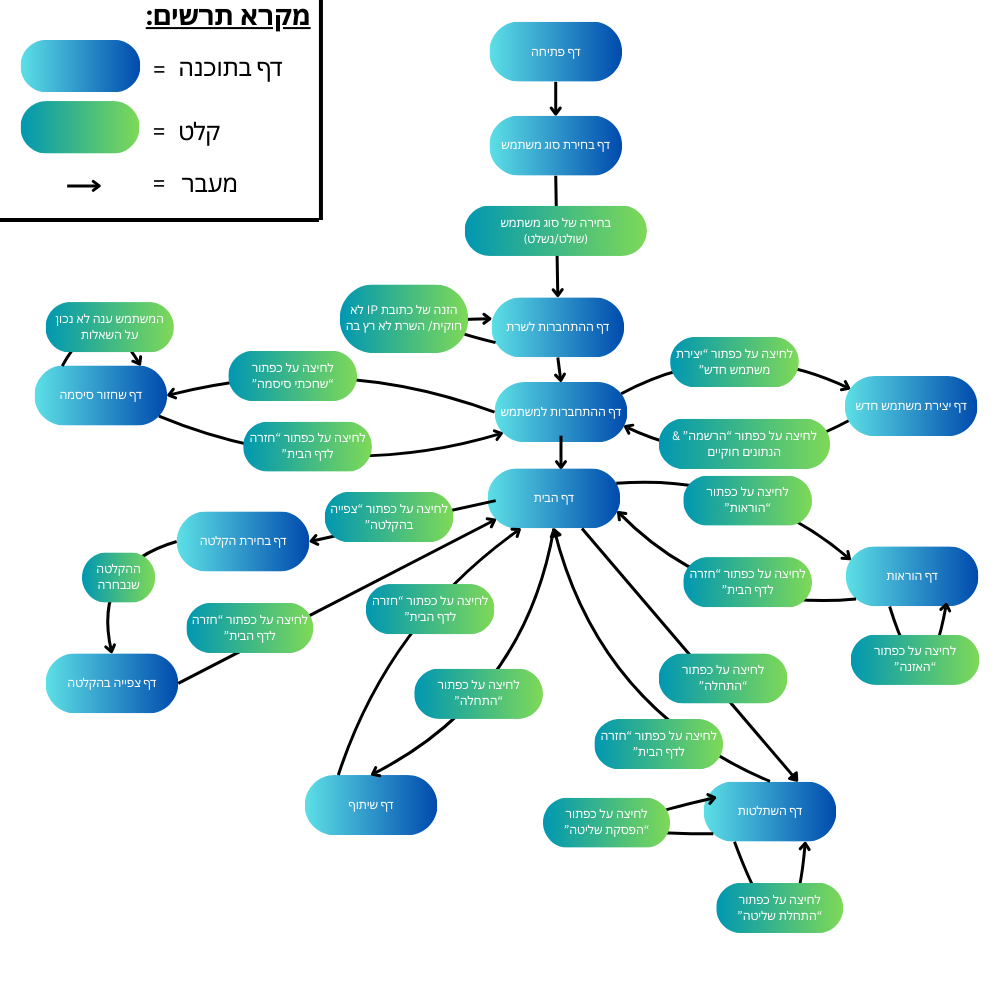
**צד שרת-**

****

**צד לקוח-**

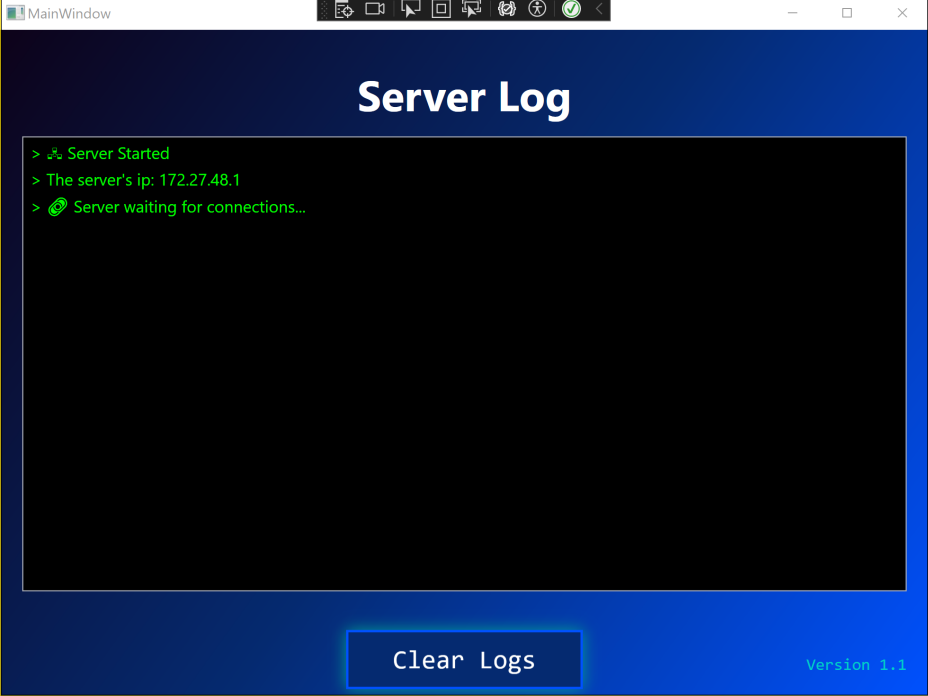
****

מדריך למשתמש

****

**הפעלת השרת:**

על מנת להתחיל ולהשתמש ב ExtremLink, יש להתחבר ראשית לשרת. כדי לעשות זאת על המשתמש להפעיל את תוכנת השרת הנקראת “ExtremLink\_Server\_v2” ולחכות שהרשת יאתחל את עצמו עד שתופיע החלונית של דף ה logs של השרת, כמו שמופיע במסך הבא:



**הפעלת הלקוח:**

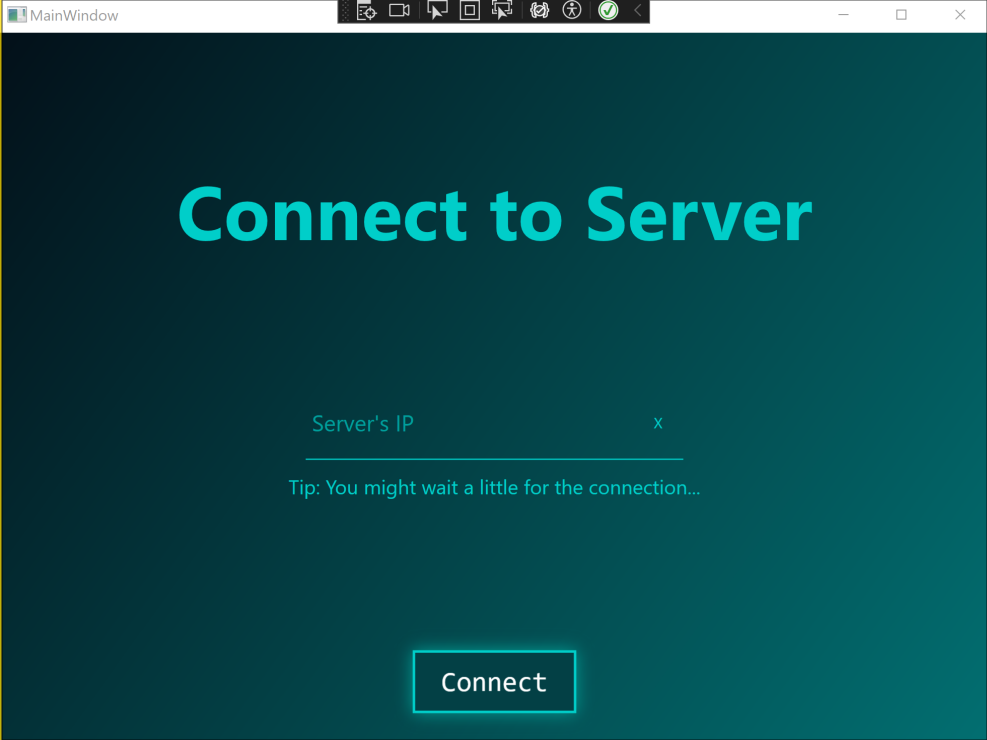
כעת, יש להפעיל את תוכנת הלקוח הנקראת “ExtremLink\_Client\_v2”. תחילה תופיעה אנימצית הפתיחה ומשם התוכנה תעבור לדף בחירת סוג המשתמש. בדף זה, על הלקוח לבחור (באמצעות לחיצה על התמונות המופיעות) האם ברצונו להיות **השולט** בלקוח אחר או זה ש**נשלט** על ידי לקוח אחר.

תמונה שמכילה טקסט, צילום מסך, גופן, גרפיקה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**התחברות הלקוח לשרת:**

לאחר שהלקוח בחר את סוג המשתמש הוא יועבר לדף ההתחברות לשרת. בדף זה, על המשתמש להזין את כתובת ה IP המקומית של המחשב שבו רץ השרת (את כתובת הזאת ניתן לראות בדף ה Logs של השרת כפי שמופיע בתחילת המדריך ב"הפעלת השרת").



**בדיקת התחברות הלקוח לשרת:**

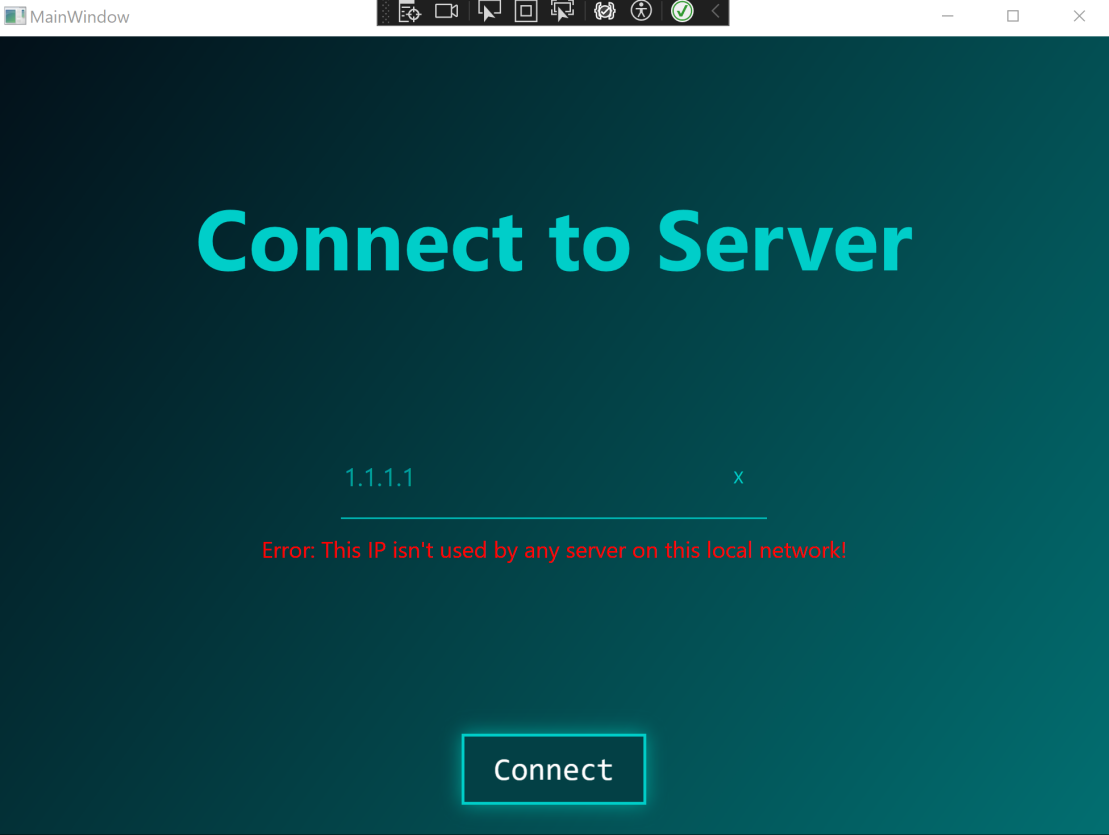
לאחר לחיצה על כפתור ה “Connect”, השרת יבדוק האם כתובת ה IP תקינה והאם ישנו שרת שמופעל על כתובת זאת כך שבמידה ולא, תוצג תוויות השגיאה הזאת בהתאם לכל סוג שגיאה:

שגיאת פורמט הכתובת:

תמונה שמכילה טקסט, צילום מסך, גופן, מותג

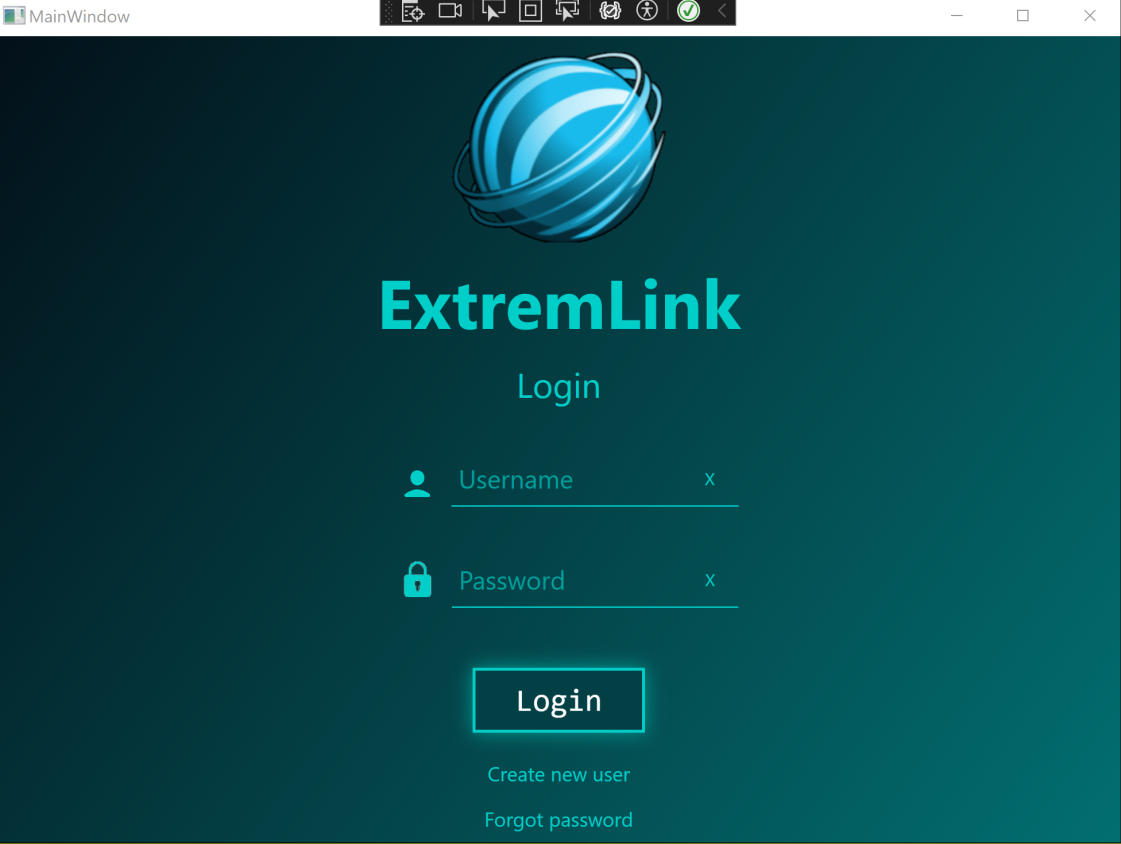
תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

שגיאת כתובת שאינה משוייכת לשרת(ברשת מקומית) :



**כניסה לחשבון קיים:**

במידה והמשתמש הינו משתמש חוזר, כלומר קיים לו חשבון, עליו להזין את שם המשתמש שלו ואת הסיסמא על מנת להיכנס לחשבון האישי שלו. לאחר מכן, יש ללחוץ על Sign In.



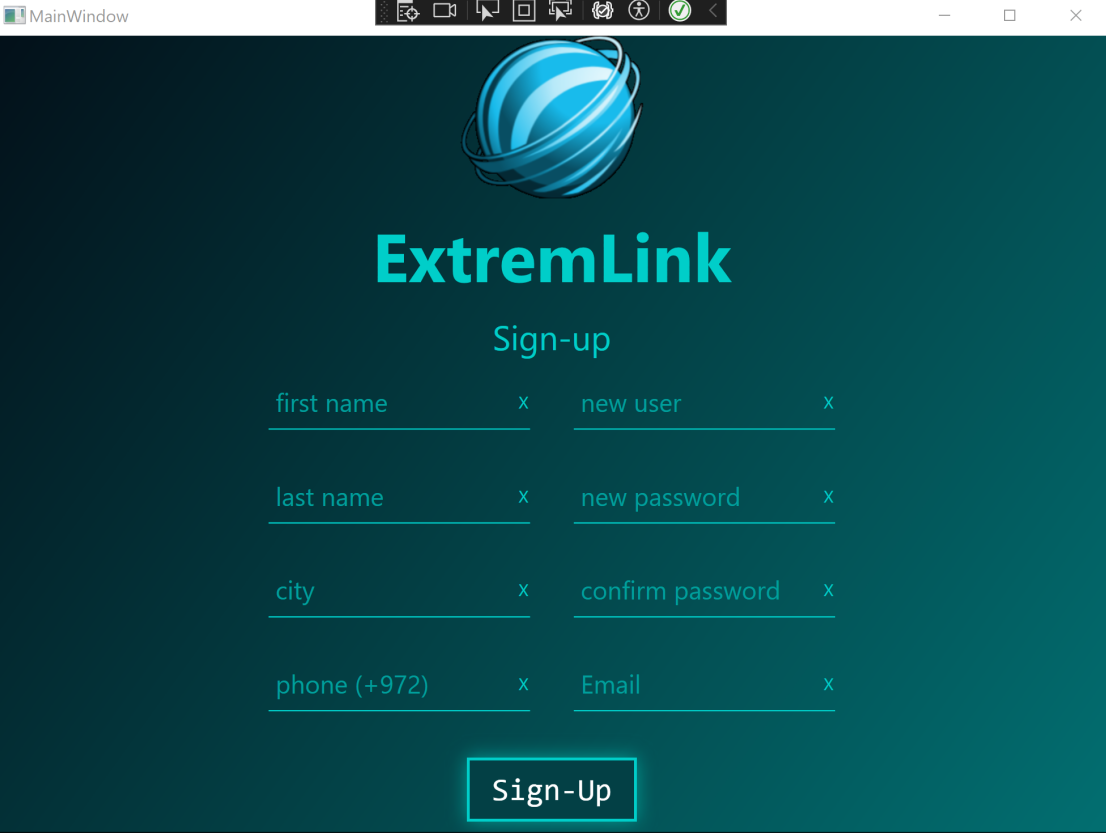
במקרה והמשתמש לא הזין נכון את נתוניו האישיים, תופיעה הודעה בהתאם ועליו לנסות שוב.

תמונה שמכילה טקסט, צילום מסך, תוכנה, עיצוב גרפי

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**יצירת חשבון חדש:**

במידה וללקוח אין משתמש עליו ליצור משתמש חדש באמצעות לחיצה על התווית התחתונה “Create new user”. לאחר מכן יפתח לו דף פתיחת החשבון הנ"ל:



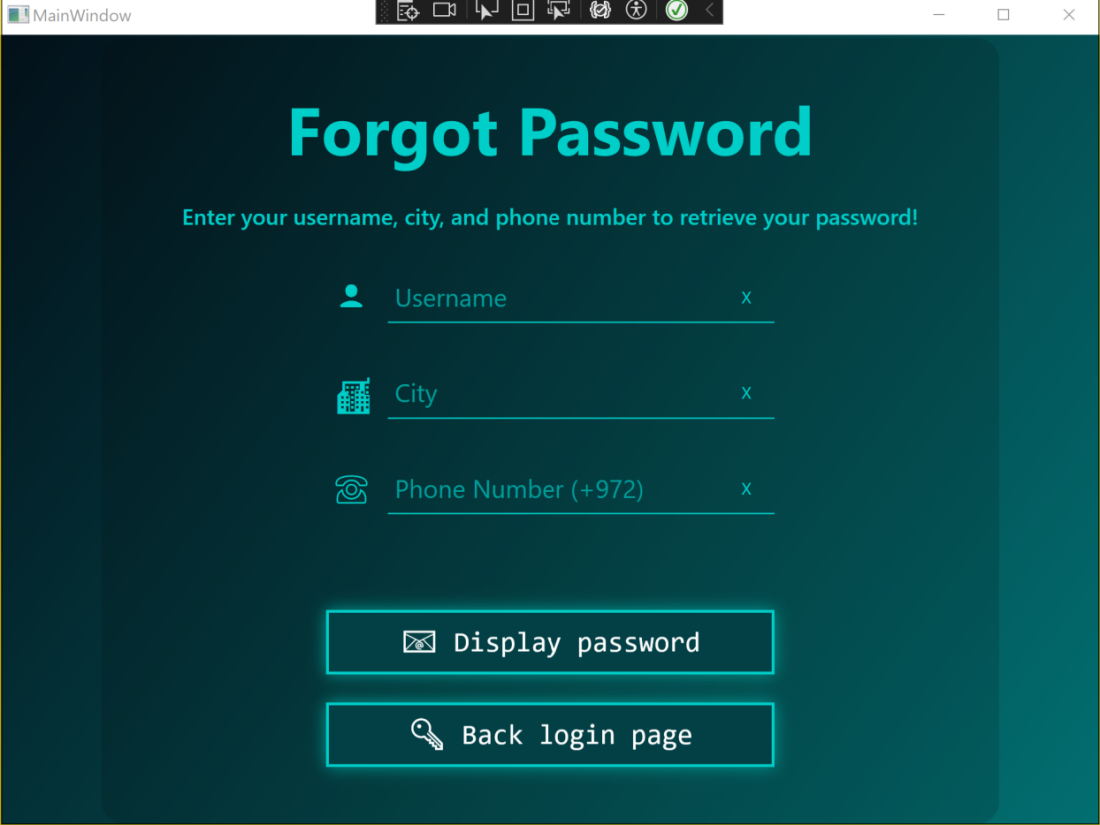
במידה והלקוח הזין נתונים שאינם תואמים את פורמט נתוני ההרשמה יופיעו הודעות שגיאה בהתאם לכל שגיאה:



לאחר שהלקוח הזין נתונים מתאימים ולחץ על כפתור ה “Sign-Up”, המערכת תיצור עבורו משתמש חדש שיכיל את אותם הנתונים במסד הנתונים שנמצא בצד השרת.

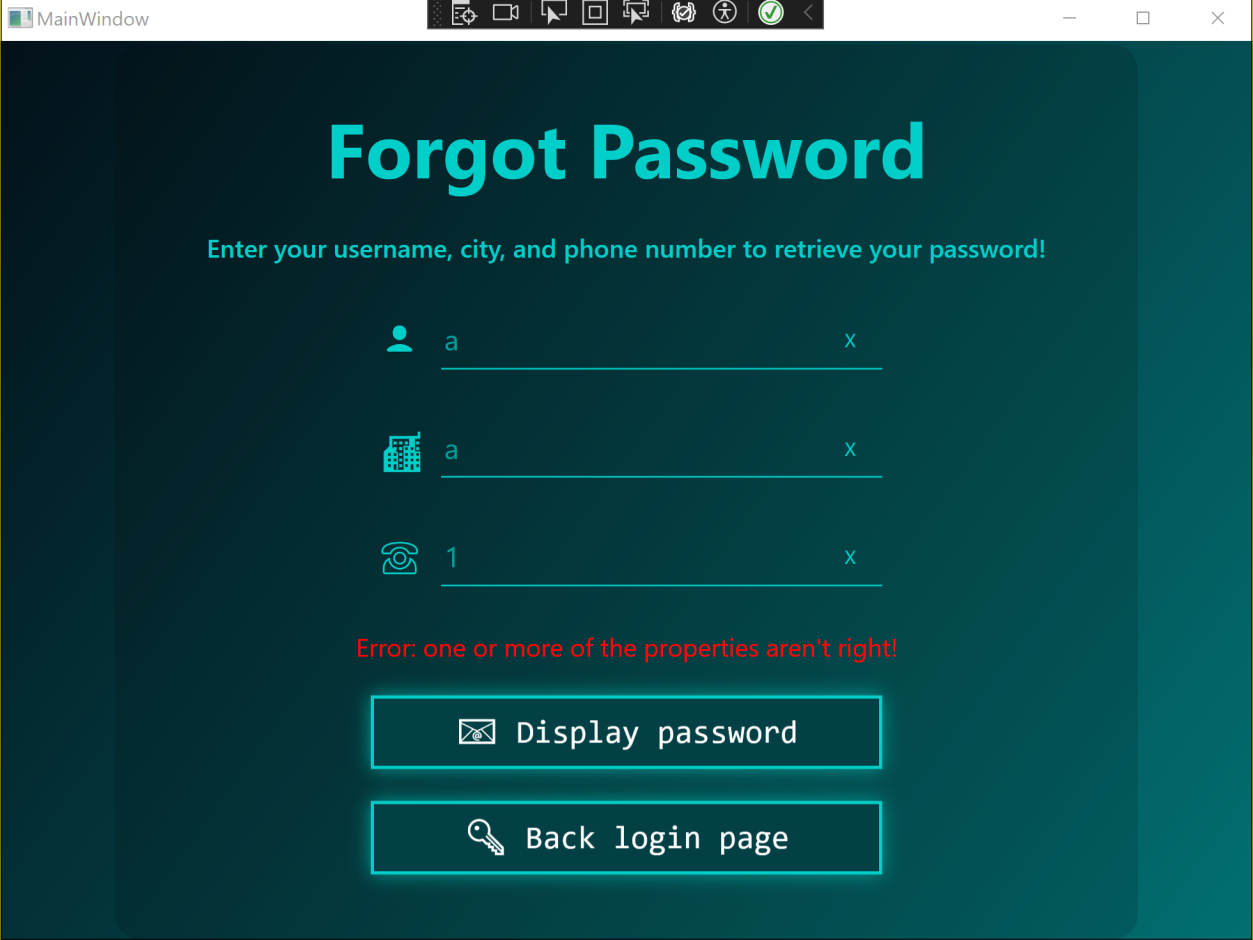
**שחזור סיסמה:**

במקרה והלקוח שכח את הסיסמה של החשבון שלו עליו ללחוץ כל תווית “Forgot password” שנמצאת בתחתית עמוד ההתחברות. הלחיצה תוביל לפתיחת דף שחזור הסיסמה:

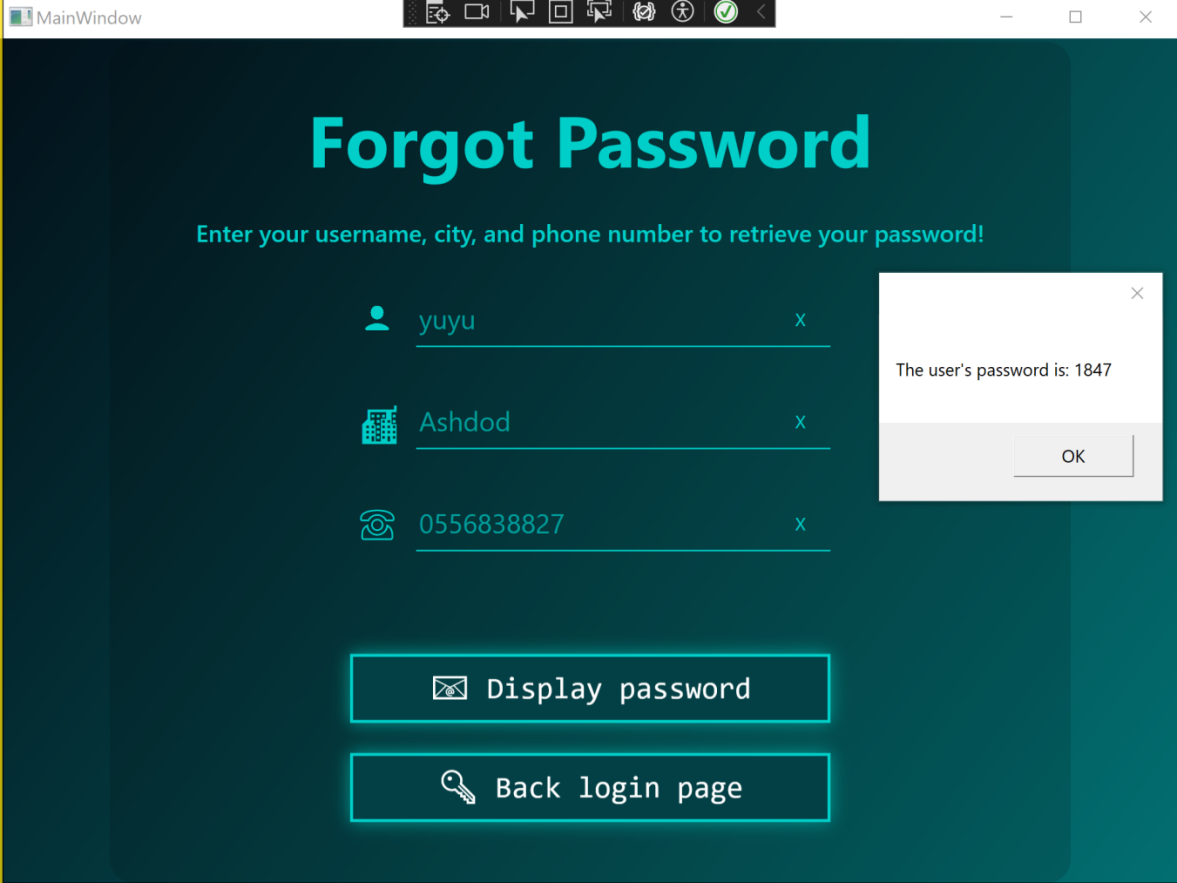


בדף זה על המשתמש לענות על שאלות מנתוני המשתמש וללחוץ על כפתור “Display password” על מנת שהלקוח יזדהה בפני המערכת כדי שהמערכת תחזיר לו את סיסמתו.

אם הלקוח טעה בהזנת הנתונים יופיעו תוויות השגיאה הבאה:

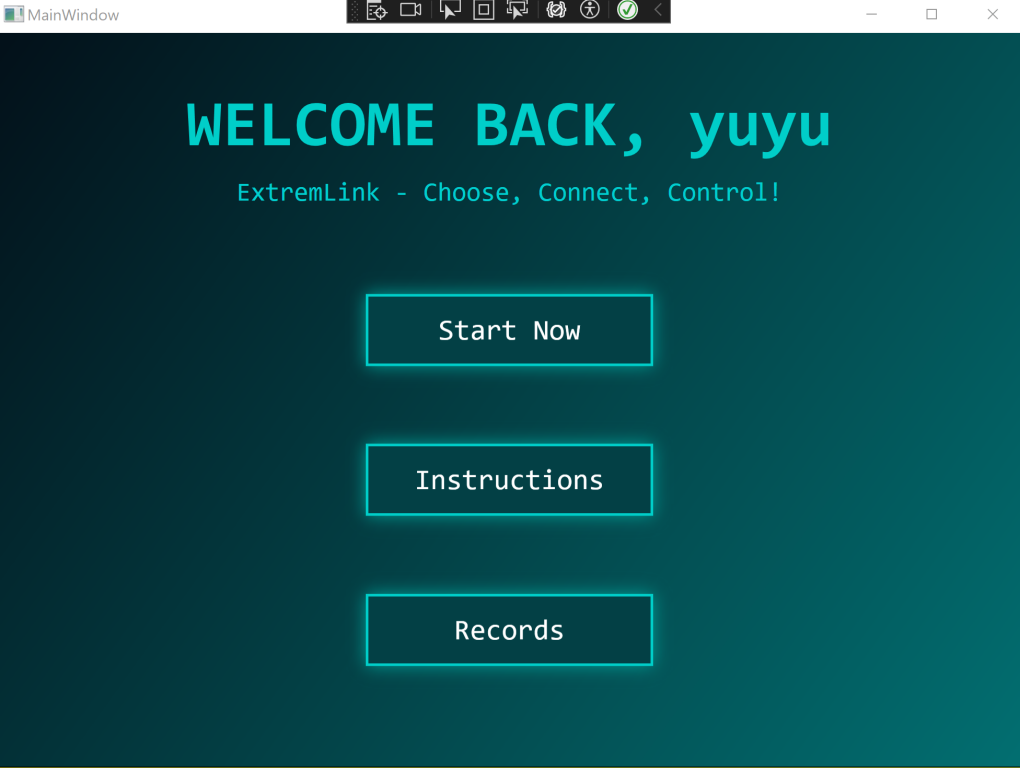


אם הלקוח ענה נכון על השאלות תופיעה הודעה המכילה את סיסמתו:



**דף הבית:**

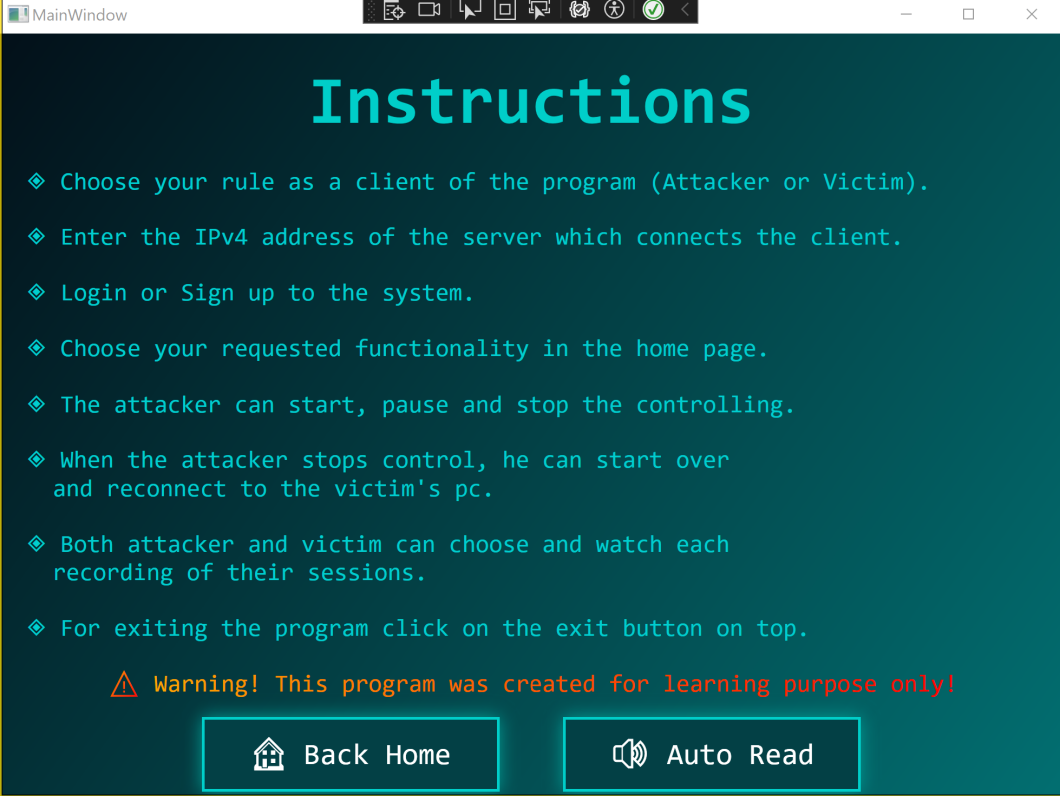
לאחר שהלקוח התחבר לשרת והתחבר למשתמש שלו הוא יועבר אוטומטית לאחר לחיצה על כפתור “Login” לדף הבית:



בדף זה המשתמש יוכל לבצע 3 פעולות עיקריות והן: כניסה לדף סשן השתלטות, כניסה לדף ההוראות וכניסה לדף בחירה וצפייה בהקלטות של סשנים.

**דף ההוראות:**

בלחיצה על כפתור “Instructions” הלקוח יועבר לדף ההוראות שבו הוא יוכל לקרוא את הוראות השימוש בתוכנה.

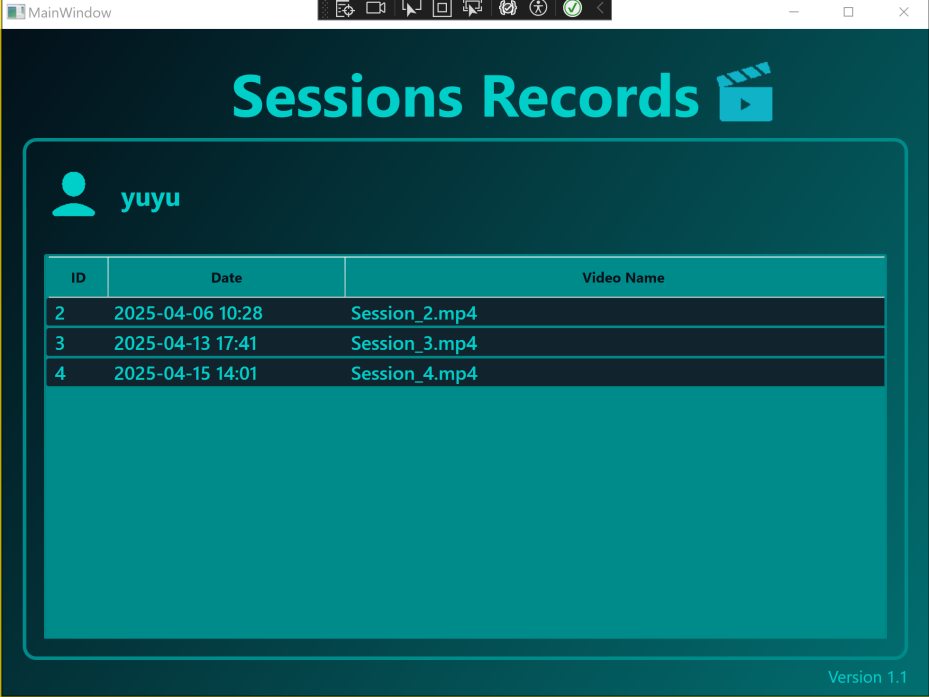


בלחיצה על כפתור “Auto Read” הלקוח יוכל להאזין להקלטת שמע של ההוראות במקום לקרוא אותן.

בלחיצה על כפתור “Back Home” הלקוח יוחזר לדף הבית של התוכנה.

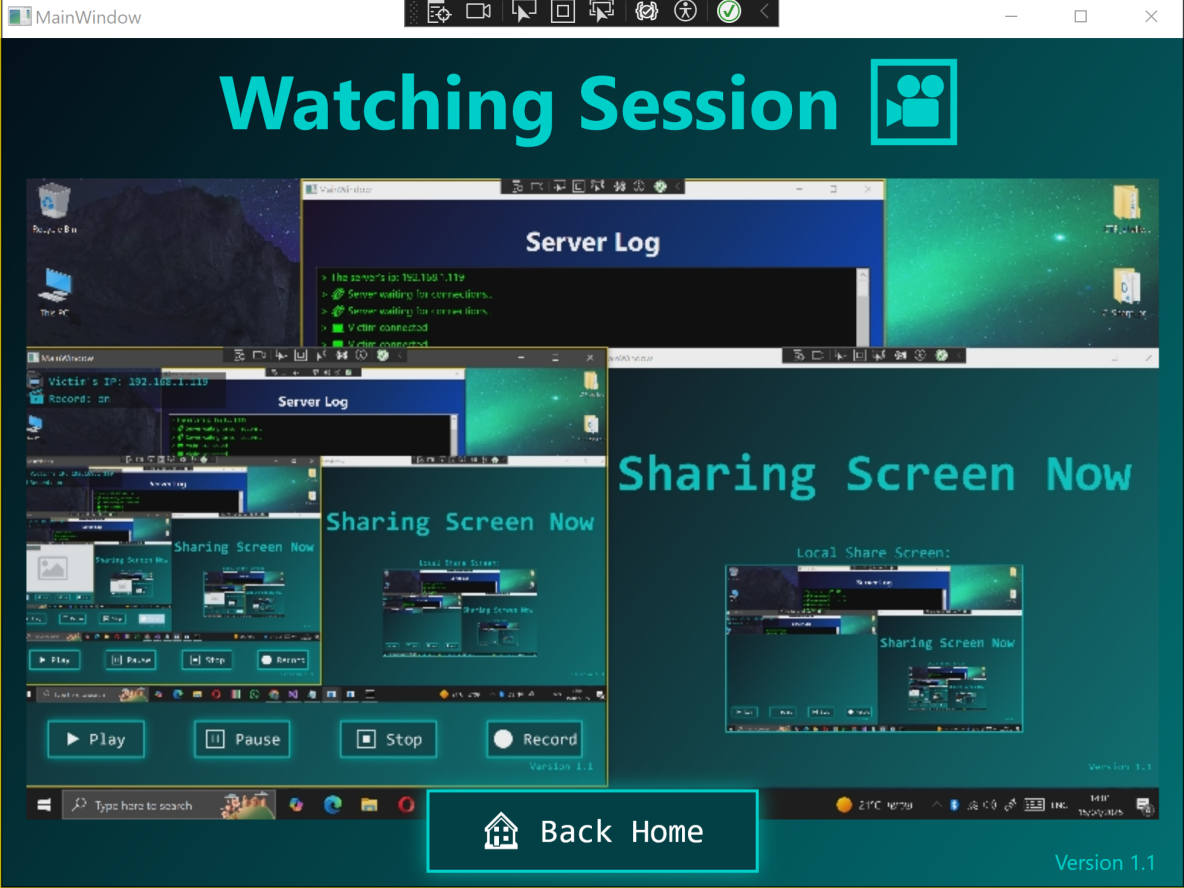
**דף בחירת הקלטה:**

בלחיצה על כפתור “Records” הלקוח יועבר לדף בחירת ההקלטות ששם תוצג בפניו רשימה של הקלטות ממסד הנתונים של שבשרת שמהן יוכל לבחור לצפות:



**דף צפייה בהקלטות:**

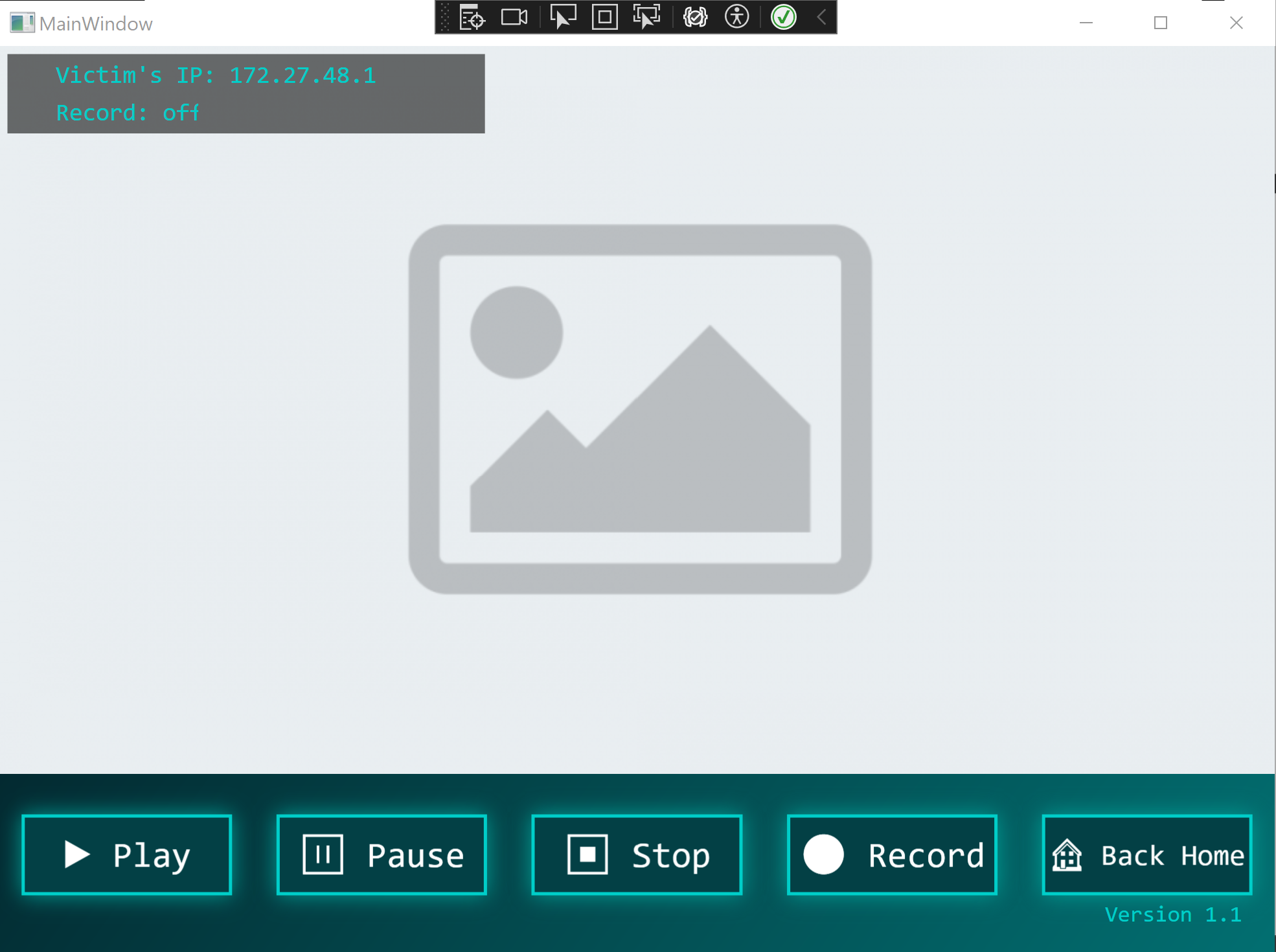
בלחיצה על אחת מהקלטות, הלקוח יועבר לדף צפייה בהקלטות שבו יוכל כמובן לצפות בהקלטה הנבחרת:



בלחיצה על כפתור “Back Home” הלקוח יוחזר לדף הבית של התוכנה.

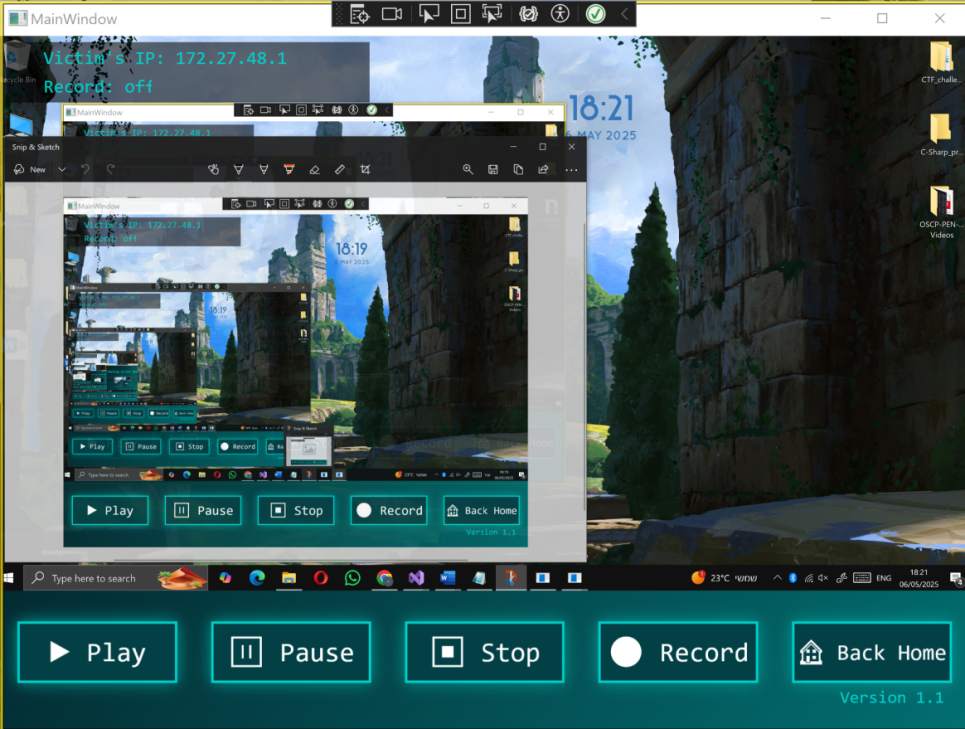
**דף השליטה:**

במידה והלקוח בחר בתפקיד השולט, לאחר לחיצה על כפתור “Start now” שמופיע בדף הבית, הלקוח יופנה לדף השליטה:



**תחילת השתלטות:**

על ידי לחיצה על כפתור “Play” הלקוח השולט יכול להתחיל לשלוט במחשבו של הלקוח הנשלט בעזרת שיתוף מסכו(פריימים של צילום מסך הלקוח), והעברת פקודות עכבר ומקלדת למחשבו של הנשלט. דוגמה לכך ניתן לראות בתמונה זו:



**הפסקת השתלטות זמנית:**

על ידי לחיצה על כפתור “Pause” הלקוח יוכל להפסיק באופן זמני את ההשתלטות שלו על המחשב הנשלט:

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**הפסקת השתלטות מוחלטת:**

על ידי לחיצה על כפתור “Stop” הלקוח לא יוכל להשתלט על מחשבו של הנתקף ולכן הוא אוטומטית יועבר לדף הבית.

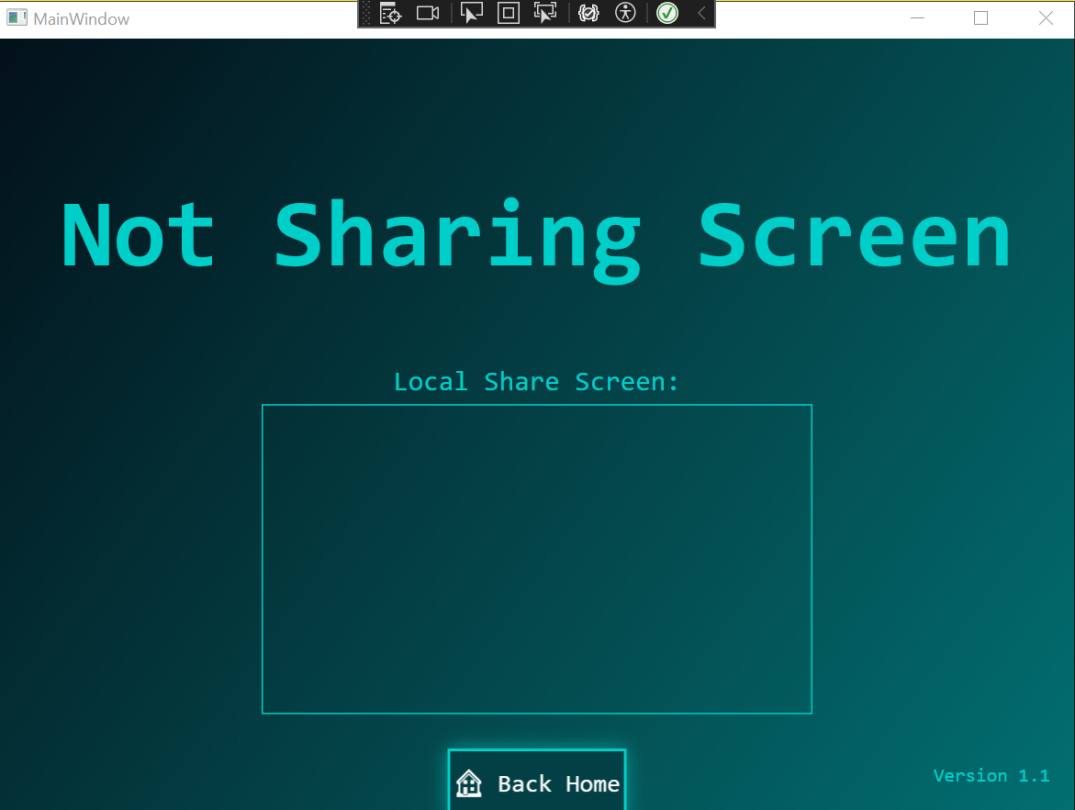
**הקלטת סשיין השתלטות:**

על ידי לחיצה על כפתור “Record” השולט למעשה מקליט את ההשתלטות שלו על מחשב התוקף ושומר את ההקלטה על שמו ובצד המסך ניתן לראות כי תווית ההקלטה השתנתה ל “on”.



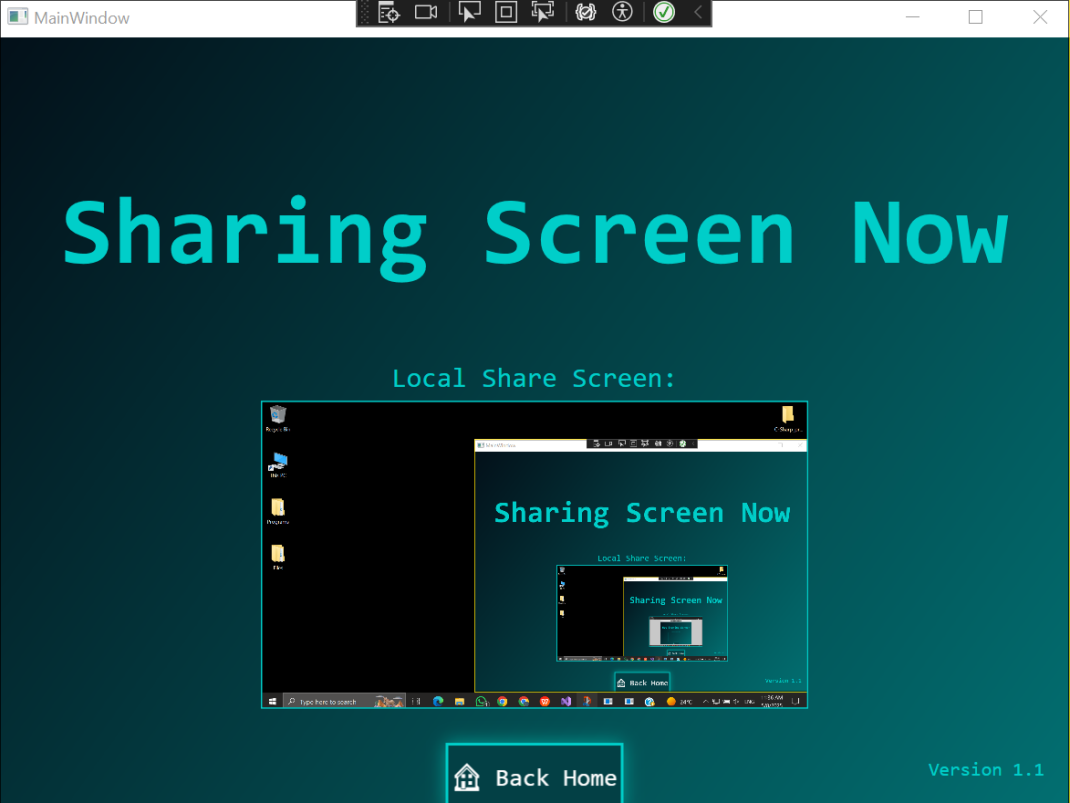
**דף שיתוף המסך:**

במידה והלקוח בחר התפקיד הנשלט ולחץ על כפתור “Start” בדף הבית, הלקוח יועבר לדף שיתוף המסך.



**שיתוף מסך**

בדף זה הלקוח ישתף את מסכו כאשר יקבל פקודה מהלקוח השולט שהוא מוכן להתחיל.



הלקוח יוכל לחזור לדף הבית כשירצה לאחר לחיצה על כפתור ה “Back Home”.

בסיסי הנתונים

**בפרוייקט זה ישנם 2 בסיסי נתונים כך שכל בסיס מכיל טבלה נפרדת:**

טבלה ראשונה

בסיס הנתונים: **ExtremLinkDB.mdf**

טבלת הנתונים: **dbo.Table**

תפקיד: **לאחסן את נתוני המשתמשים של המערכת ולשלוף בעת הצורך את הנתונים.**

|  |  |  |  |
| --- | --- | --- | --- |
| תכונות נוספות | האם הוא חובה? | טיפוס הנתונים | שם העמודה |
|  | **כן** | int | **Id** |
| **מפתח ראשי** | **כן** | Nvarchar(50) | **username** |
|  | **כן** | Nvarchar(50) | **password** |
|  | **לא** | Nvarchar(50) | **firstName** |
|  | **לא** | Nvarchar(50) | **lastName** |
|  | **לא** | Nvarchar(50) | **city** |
|  | **לא** | Nvarchar(50) | **phone** |
|  | **לא** | Nvarchar(50) | **email** |

טבלה שנייה

בסיס הנתונים: **SessionRecordsDB.mdf**

טבלת הנתונים: **dbo.Table**

תפקיד: **לאחסן הקלטות של סשינים של השתלטויות של השולט על הנשלט (ההקלטות משוייכות לשולט) ואיחזור שלהן כשצריך.**

|  |  |  |  |
| --- | --- | --- | --- |
| תכונות נוספות | האם הוא חובה? | טיפוס הנתונים | שם העמודה |
| **מפתח ראשי** | **כן** | int | **Id** |
|  | **כן** | Nvarchar(50) | **Username** |
|  | **כן** | Varbinary(MAX) | **VideoData** |
|  | **כן** | datetime | **RecordedTime** |

**שמירה ואחזור הנתונים מטבלאות הנתונים:**

בפרוייקט, במחלקות Server ו Session שנמצאות בצד השרת מצויים הפונקציות שאחראיות לשמירה ואחזור נתונים מטבלאות הנתונים שבמסדי הנתונים.

הוספה של משתמש:

public bool AddUser(string[] parameters, string databaseFileName)

הוספה של הקלטה:

public void UploadSessionToDatabase()

אחזור של ערך בולאיני לגבי האם משתמש קיים:

public bool IsUserExist(string username, string city, string phone, string databaseFileName)

public bool IsUserExist(string username, string password, string databaseFileName)

אחזור של סיסמאת משתמש:

public string RetrievePassword(string username, string city, string phone, string databaseFileName)

אחזור של הקלטה:

private byte[] GetSessionContentById(int id)

**מדריך למפתח**

**שרת:**

מחלקת **Server**

**מחלקת Server אחראית על ניהול התקשורת בין לקוחות השולט והנשלט בתכונה. היא מטפלת בהתחברות ויצירת הלקוחות, בהעברת פקודות וידאו, מקלדת ועכבר, ובניהול משתמשים והקלטות מול מסד הנתונים, תוך התבססות על תבנית העיצוב Singleton.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| בנאי של המחלקה שיוצר אובייקט של המחלקה ומאתחל את התכונות של המחלקה. | private Server() |
| הפונקציה מחזירה את כתובת ה **IPv4** של המחשב שעליו רצה הפונקציה. | private string FindIpAddress() |
| הפונקציה מריצה 2 **משימות א-סינכרוניות** לחיבור 2 הלקוחות לשרת. | public void ConnectToClients() |
| הפונקציה מאזינה לחיבורים מהלקוח **השולט** ומתחברת אליו ברגע שהוא שולח בקשת התחברות. בנוסף היא גם מריצה 2 **משימות א-סינכרוניות** לניהול הפרוטוקול בעבור 2 סוקטים של התוקף המבוססים פרוטוקול **TCP**. | private async Task ConnectToAttacker() |
| הפונקציה מאזינה לחיבורים מהלקוח **הנשלט** ומתחברת אליו ברגע שהוא שולח בקשת התחברות. בנוסף היא גם מריצה 2 **משימות א-סינכרוניות** לניהול הפרוטוקול בעבור 2 סוקטים של הנתקף המבוססים פרוטוקול **TCP**. | private async Task ConnectToVictim() |
| הפונקציה שולחת את כתובת ה **IP** של 2 הלקוחות אחד לשני כדי שכל לקוח ידע מה ה **IP** של הלקוח השני. | public void UpdateClientsIPAddress() |
| הפונקציה מתנהלת באופן **א-סינכרוני** עם ההודעות ש 2 הלקוחות שולחים לשרת על גבי סוקט ה **UDP**. | private async Task HandleUdpCommunication() |
| הפונקציה מתנהלת באופן **א-סינכרוני** עם ההודעות שהלקוח **השולט** שולח לשרת על גבי סוקט ה **TCP** הרגיל. | private async Task HandleAttackerTcpCommunication() |
| הפונקציה מתנהלת באופן **א-סינכרוני** עם ההודעות שהלקוח **הנשלט** שולח לשרת על גבי סוקט ה **TCP** הרגיל. | private async Task HandleVictimTcpCommunication() |
| הפונקציה מתנהלת באופן **א-סינכרוני** עם ההודעות שהלקוח **השולט** שולח לשרת על גבי סוקט ה **TCP** של ההקלטות. | private async Task HandleAttackerSessionTcpCommunication() |
| הפונקציה מקבלת הודעה של **בקשת התחברות** ותפקידה לשלוח ללקוח הודעה בהתאם בנוגע לאם האימות הצליח או לא. | private void HandleLoginMessages(string data, TypeOfClient typeOfClient) |
| הפונקציה מוסיפה **למסד הנתונים** של המשתמשים משתמש חדש לפי הנתונים שנכללו בהודעה שהפונקציה קיבלה. | private void HandleSignUpMessage(string data, TypeOfClient typeOfClient) |
| הפונקציה מקבלת את הנתונים שלקוח ששכח **סיסמה** סיפק לשרת על מנת **לשחזר את סיסמתו** ולמעשה מבצעת אימות עקיף עם המערכת על סמך נתונים אחרים כך שהיא שולחת את הסיסמה ללקוח אם האימות הצליחה, אחרת הודעת שגיאה. | private void HandleForgotPassword(string data, TypeOfClient typeOfClient) |
| הפונקציה מתנהלת עם הודעות מסוג "**ניהול משתמשים**". | private void HandleUsersDatabaseMessages(string data, TypeOfClient typeOfClient) |
| הפונקציה מתנהלת עם הודעות פקודה מהלקוח **השולט** בנוגע **לתפעול השליטה** של שיתוף המסך על הלקוח **הנשלט** בכך שהיא מעבירה את ההודעה ללקוח **הנשלט**. | private void HandleFramesMessages(string data, TypeOfClient typeOfClient) |
| הפונקציה מתנהלת עם הודעות פקודה מהלקוח **השולט** בנוגע **לתפעול השליטה** בעכבר של הלקוח **הנשלט** בכך שהיא מעבירה את ההודעה ללקוח **הנשלט**. | private void HandleMouseMessages(string data) |
| הפונקציה מתנהלת עם הודעות פקודה מהלקוח **השולט** בנוגע **לתפעול השליטה** במקלדת של הלקוח **הנשלט** בכך שהיא מעבירה את ההודעה ללקוח **הנשלט**. | private void HandleKeyboardMessages(string data) |
| הפונקציה מקבלת שם של לקוח **השולט** ומחזירה **רשימת** **IList** שמכילה את נתוני כל ההקלטות של **השולט**. | private IList<Session> UsersSessionsProperties(string username) |
| הפונקציה **ממירה** **רשימת** **IList** שמכילה נתוני הקלטות לפורמט **Json**. | private string FromSessionIlistToStringJson(IList<Session> sessionIlist) |
| הפונקציה מחזירה **מערך בתים** המכיל את התוכן של ההקלטות עצמן. | private byte[] GetSessionContentById(int id) |
| הפונקציה שולחת את התוכן הבינארי של ההקלטות עצמן ללקוחות שביקשו לצפות בהן. | private void SendSessionContent(byte[] sessionContent, Socket socket) |
| הפונקציה מתנהלת עם הודעות מסוג **הקלטות** שהלקוח שלח לשרת בכך שהיא מספקת לשולח את **המידע בנוגע להקלטותיו** בהתאם לבקשתו. | private void HandleSessionInfoMessage(string data) |
| הפונקציה מקבלת תוכן בינארי של הפריים שהלקוח הנשלט שלח לשרת ומחזירה אותו כמערך של בתים. | public byte[] GetFrame() |
| הפונקציה שולחת את התוכן הבינארי של הפריים, שהלקוח הנשלט שלח לשרת, ללקוח השולט. | public void SendFrame(byte[] frame) |
| הפונקציה מקבלת תוכן בינארי של הקלטה שהלקוח השולט שלח לשרת ומחזירה אותו כמערך של בתים. | public byte[] GetSession() |
| הפונקציה מחזירה אובייקט של SqlConnection משמש לפתיחת חיבור למסד נתונים של השרת, לפי השם של מסד הנתונים שהתקבל בפונקציה, כדי לאפשר שליחה ושליפה של נתונים מהמסד באמצעות שאילתות. | public static SqlConnection ConnectToDB(string fileName) |
| הפונקציה מחזירה true או false בנוגע להאם ישנו משתמש בעל שם המשתמש והסיסמה שהפונקציה קיבלה במסד הנתונים הנתון. | public bool IsUserExist(string username, string password, string databaseFileName) |
| הפונקציה מחזירה true או false בנוגע להאם ישנו משתמש בעל השדות של העיר, הטלפון ושם המשתמש שהפונקציה קיבלה במסד הנתונים הנתון. | public bool IsUserExist(string username, string city, string phone, string databaseFileName) |
| הפונקציה מקבלת שדות של שם משתמש, עיר, טלפון ושם מסד הנתונים ומחזירה את הסיסמה המשויכת לשם משתמש זה. במידה ואין משתמש כזה הפונקציה מחזירה הודעה בהתאם. | public string RetrievePassword(string username, string city, string phone, string databaseFileName) |
| הפונקציה מקבלת פרמטרים ליצירת משתמש חדש ומוסיפה אותו למסד הנתונים הנתון. במידה והפעולה צלחה הפונקציה מחזירה true, אחרת false. | public bool AddUser(string[] parameters, string databaseFileName) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה מטיפוס מחרוזת שמכיל את כתובת ה IP של השרת. | private string serverIpAddress; |
| משתנה מטיפוס מחזורת שמכיל את תגובת השרת. | private string respond; |
| משתנה מטיפוס Client שמייצג את הלקוח השולט. | private Client attacker; |
| משתנה מטיפוס Client שמייצג את הלקוח הנשלט. | private Client victim; |
| משתנה מטיפוס Session שמייצג את ההקלטה הנוכחית שהלקוח השולט שלח לשרת. | private Session tempSession; |
| אובייקט סטטי מטיפוס Server שמייצג אובייקט יחידני של מחלקת ה Sever ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static Server serverInstance; |

מחלקת **Client**

**מחלקת Client מייצגת לקוח (שולט או נשלט) בצד השרת. היא אחראית על ניהול החיבורים מסוג TCP ו-UDP מול השרת, קבלת ושליחת הודעות, שמירת פרטי המשתמש וה-IP של הלקוח, ותמיכה בתקשורת עבור זרמי וידאו וסשנים מוקלטים. המחלקה מהווה את נקודת הקצה בתקשורת מצד השרת מול הלקוח הרלוונטי.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| בנאי של המחלקה שיוצר אובייקט של המחלקה ומאתחל את התכונות של המחלקה. | public Client(string serverIpAddress, TypeOfClient typeOfClient) |
| פונקציה שמשנה את הגבלת רוחב הפס של אובייקט הסוקט שהתקבל. | private void ChangeSocketBufferSizeLimit(Socket socket, int RecieveBufferSize = 8192, int SendBufferSize = 8192) |
| פונקציה שמחזירה את כתובת ה IP של הלקוח שהתחבר לשרת כמחרוזת. | private string FindClientIpAddress(Socket clientSocket) |
| פונקציה שמאזינה וממתינה לחיבור של הלקוחות לשרת (לסוקטים מסוג TCP של כל לקוח). | public async Task ConnectToClient() |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ואובייקט סוקט ושולחת הודעה ללקוח לפי מבנה מוגדר מראש שפרוטוקול התוכנה פועל לפיו. | private void SendMessageToClient(string message, string typeOfMessage, Socket socket) |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ושולחת את ההודעה ללקוח דרך הסוקט מסוג TCP הרגיל של הלקוחות. | public void SendTCPMessageToClient(string typeOfMessage, string message) |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ושולחת את ההודעה ללקוח דרך הסוקט מסוג UDP של הלקוחות. | public void SendUDPMessageToClient(string typeOfMessage, string message) |
| פונקציה שאחראית להוציא את המידע מההודעה לפי פורמט הפרוטוקול ולהחזירו בצורה מסודרת ונגישה ברשימה. | private List<object> OrderMessage(string message) |
| פונקציה שאחראית לקבל הודעה מסוקט ה TCP הרגיל של הלקוח, להמיר את ההודעה מטיפוס מערך בתים למחרוזת ולהחזיר את תוכן ההודעה ומאפייניה ברשימה. | public List<object> GetTCPMessageFromClient() |
| פונקציה שאחראית לקבל הודעה מסוקט ה UDP של הלקוח, להמיר את ההודעה מטיפוס מערך בתים למחרוזת ולהחזיר את תוכן ההודעה ומאפייניה ברשימה. | public List<object> GetUDPMessageFromClient() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| קבוע שמייצג את מספר הפורט של סוקט ה TCP הרגיל של הלקוח השולט. | public readonly int ATTACKER\_TCP\_PORT; |
| קבוע שמייצג את מספר הפורט של סוקט ה UDP של הלקוח השולט. | public readonly int ATTACKER\_UDP\_PORT; |
| קבוע שמייצג את מספר הפורט של סוקט ה TCP של ההקלטות של הלקוח השולט. | public readonly int ATTACKER\_SESSION\_TCP\_PORT; |
| קבוע שמייצג את מספר הפורט של סוקט ה TCP הרגיל של הלקוח הנשלט. | public readonly int VICTIM\_TCP\_PORT; |
| קבוע שמייצג את מספר הפורט של סוקט ה UDP של הלקוח הנשלט. | public readonly int VICTIM\_UDP\_PORT; |
| קבוע שמייצג את מספר הפורט של סוקט ה TCP של ההקלטות של הלקוח הנשלט. | public readonly int VICTIM\_SESSION\_TCP\_PORT; |
| משתנה מטיפוס מחרוזת שאחראי לשמור את כתובת ה IP של השרת. | private string serverIpAddress; |
| משתנה מטיפוס מחרוזת שאחראי לשמור את כתובת ה IP של הלקוח. | private string ipAddress; |
| משתנה מסוג סוקט מסוג TCP שמנהל את התקשורת בין השרת ללקוח (בהקשר להודעות שאינן הקלטות). | private Socket tcpSocket; |
| משתנה מסוג סוקט מסוג UDP שמנהל את התקשורת בין השרת ללקוח (בהקשר לפריימים של שיתוך המסך). | private Socket udpSocket; |
| משתנה מסוג סוקט מסוג TCP שמנהל את התקשורת בין השרת ללקוח (בהקשר להקלטות). | private Socket sessionTcpSocket; |
| משתנה שמייצג נקודת קצה של הלקוח בצד השרת (IP ופורט, בשונה מהסוקט, הוא מכיל את הפורט החיצוני של הסוקט). | private IPEndPoint clientUdpEndPoint; |
| משתנה מטיפוס מחרוזת ששומר את שם המשתמש של הלקוח. | private User user; |
| משתנה מסוג TypeOfClient (Enum שמעיד על סוג הלקוח: שולט/נשלט). | private TypeOfClient typeOfClient; |
| משתנה מטיפוס בוליאני שמעיד על חיבורו או אי חיבורו של השרת ללקוח (true מחובר, false לא מחובר). | private bool isConnected; |

מחלקת **Session**

**מחלקת Session מייצגת סשן של שליטה מרחוק הכולל הקלטת וידאו של פעילות המשתמש. היא שומרת את תוכן הווידאו כ־byte array, את זמן ההקלטה, את שם המשתמש ואת מזהה הסשן (ID). בנוסף, היא מאפשרת להעלות את הסשן למסד הנתונים לצורכי אחסון ושימוש מאוחר יותר.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| בנאי של המחלקה שמקבל את זמן ההקלטה ואת שם המשתמש של הלקוח השולט בהקלטה. | public Session(DateTime recordedTime, string username) |
| בנאי של המחלקה שמקבל את זמן ההקלטה, את שם המשתמש של הלקוח השולט בהקלטה ואת המספר המזהה של ההקלטה. | public Session(DateTime recordedTime, string username, int id) |
| הפונקציה מחזירה אובייקט של SqlConnection משמש לפתיחת חיבור למסד נתונים של השרת, למסד הנתונים של ההקלטות, כדי לאפשר שליחה ושליפה של נתונים מהמסד באמצעות שאילתות. | private SqlConnection ConnectToDB() |
| פונקציה שאחראית להעלות את נתוני אובייקט ההקלטה למסד הנתונים. | public void UploadSessionToDatabase() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| פרמטר מטיפוס DateTime(סוג פרמטר ששומר ערכי זמן) ששומר את זמן הקלטת ההקלטה. | private DateTime recordedTime; |
| פרמטר מטיפוס מערך של בתים שמאחסן את ההקלטה עצמה. | private byte[] videoContent; |
| משתנה מטיפוס מחרוזת ששומר את שם המשתמש של מבצע ההקלטה (הלקוח השולט). | private string username; |
| פרמטר מטיפוס מספר שלם ששומר את מספר המזהה של ההקלטה. | private int id; |

מחלקת **Log**

**המחלקה Log מנהלת יומן רישום (Log) של הודעות בצד השרת, באמצעות שמירה של המחלקה Log מנהלת יומן רישום (Log) של הודעות בצד השרת, באמצעות שמירה של רשימת הודעות תוך שימוש בתבנית העיצוב Singleton, המחלקה מאפשרת הוספה של הודעות ליומן ואיחסונם.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | private Log() |
| פונקציה שמוסיפה הודעה מטיפוס מחרוזת לרשימת ההודעות של יומן ה Log של השרת. | public void AddMessage(string message) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| אובייקט מטיפוס IList שמהווה רשימה ששומרת הודעות של יומן ה Log של השרת. | private IList<string> messages; |
| אובייקט סטטי מטיפוס Log שמייצג אובייקט יחידני של מחלקת ה Log ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static Log logInstance; |

מחלקת **User**

**מחלקת User מייצגת משתמש של לקוח במערכת. היא שומרת את שם המשתמש ומספקת ממשק פשוט לגישה למידע זה. המחלקה משמשת כתשתית בסיסית לזיהוי וניהול משתמשים בצד השרת.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | public User() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה ששומר את שם המשתמש של הלקוח. | private string username; |

**לקוח:**

מחלקת **Client**

מחלקת **Client** מייצגת לקוח שמתחבר לשרת מרחוק. היא אחראית על יצירת והגדרת החיבורים בין המחשב של המשתמש לשרת, כולל חיבורים רגילים וחיבורים מיוחדים לפעולות ספציפיות. המחלקה מאפשרת קבלת פרטים מהשרת כמו כתובתו והודעות וכמו כן אחראית לשלוח אליו הודעות ספציפיות לשרת וללקוחות אחרים דרך השרת.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | public Client() |
| פונקציה שמשנה את הגבלת רוחב הפס של אובייקט הסוקט שהתקבל. | private void ChangeSocketBufferSizeLimit(Socket socket, int RecieveBufferSize=8192, int SendBufferSize=8192) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה מסוג סוקט מסוג TCP שמנהל את התקשורת בין השרת ללקוח (בהקשר להודעות שאינן הקלטות). | protected Socket tcpSocket; |
| משתנה מסוג סוקט מסוג UDP שמנהל את התקשורת בין השרת ללקוח (בהקשר לפריימים של שיתוך המסך). | protected Socket udpSocket; |
| משתנה מסוג סוקט מסוג TCP שמנהל את התקשורת בין השרת ללקוח (בהקשר להקלטות). | protected Socket sessionTcpSocket; |
| משתנה מטיפוס מחרוזת שאחראי לשמור את כתובת ה IP של השרת. | protected string serverIpAddr; |
| משתנה מטיפוס מחרוזת שאחראי לשמור את תגובת השרת לפי הודעותיו. | protected string serverRespond; |
| משתנה שמייצג נקודת קצה של השרת בצד הלקוח (IP ופורט, בשונה מהסוקט, הוא מכיל את הפורט החיצוני של הסוקט). | protected IPEndPoint serverUdpEndPoint; |

מחלקת **Attacker**

מחלקת **Attacker** מייצגת תוקף המחובר לשרת מרחוק ומנהל תקשורת עם קורבן. היא יורשת ממחלקת Client ומתמקדת בשליטה ובקרה על הקורבן - כולל קבלת תמונות מסך בזמן אמת, ניהול הקלטות, וטיפול בפקודות מקלדת ועכבר. המחלקה משתמשת בתבנית העיצוב **Singleton** ומטפלת בשליחה וקבלה של הודעות דרך חיבורי רשת ייעודיים, תוך פיצול הנתונים לחבילות והרכבתם מחדש.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. פונקציה זו קוראת ממחלקת האב לפונקציית הבנאי של מחלקת Client. | private Attacker() : base() |
| פונקציה ששולחת לשרת בקשת התחברות דרך הסוקטים מסוג TCP של הלקוח השולט. | public void ConnectToServer(string serverIpAddr) |
| פונקציה שמריצה באופן א-סינכרוני את הפונקציות שמנהלות את הפרוטוקול של התוכנה בעבור כל סוקט של הלקוח התוקף. | public void Start() |
| פונקציה שמקבלת מחרוזות של הודעה, סוג הודעה וסוקט ושולחת את ההודעה ללקוח דרך הסוקט אל השרת. | private void SendMessageToClient(string typeOfMessage, string message, Socket socket) |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ושולחת את ההודעה לשרת דרך הסוקט מסוג TCP הרגיל של הלקוח השולט. | public void SendTCPMessageToClient(string typeOfMessage, string message) |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ושולחת את ההודעה לשרת דרך הסוקט מסוג UDP של השרת. | public void SendUDPMessageToClient(string typeOfMessage, string message) |
| פונקציה שאחראית להוציא את המידע מההודעה לפי פורמט הפרוטוקול ולהחזירו בצורה מסודרת ונגישה ברשימה. | private List<object> OrderMessage(string message) |
| פונקציה שאחראית לקבל הודעה מסוקט ה TCP הרגיל של השרת, להמיר את ההודעה מטיפוס מערך בתים למחרוזת ולהחזיר את תוכן ההודעה ומאפייניה ברשימה. | public List<object> GetTCPMessageFromClient() |
| פונקציה שאחראית לקבל הודעה מסוקט ה UDP של השרת, להמיר את ההודעה מטיפוס מערך בתים למחרוזת ולהחזיר את תוכן ההודעה ומאפייניה ברשימה. | public List<object> GetUDPMessageFromClient() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהשרת ללקוח השולט דרך הסוקט מסוג TCP הרגיל. | private async Task HandleTcpCommunication() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהרשת ללקוח השולט דרך הסוקט מסוג UDP. | private async Task HandleUdpCommunication() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהשרת ללקוח השולט דרך הסוקט מסוג TCP של ההקלטות. | private void HandleGeneralStuffMessages(string data) |
| פונקציה שמתנהלת עם הודעות מסוג ניהול משתמשים שהשרת שלח ללקוח השולט. | private void HandleUsersManagmentCommands(string data) |
| פונקציה שמתנהלת עם הודעות מסוג שיתוף מסך שהשרת שלח ללקוח השולט. | private void HandleFramesCommands(string data) |
| פונקציה שמתנהלת עם הודעות מסוג פקודות עכבר שהשרת שלח ללקוח השולט. | private void HandleMouseInput(string message) |
| פונקציה שמתנהלת עם הודעות מסוג פקודות מקלדת שהשרת שלח ללקוח השולט. | private void HandleKeyboardInput(string message) |
| פונקציה שמתנהלת עם הודעות מסוג ניהול הקלטות שהשרת שלח ללקוח השולט. | private void HandleSessionsCommands(string message) |
| פונקציה שמקבלת הקלטה מהשרת בסוקט מסוג TCP האחראי על ההקלטות ומחזיר את ההקלטה כמערך של בתים. | private byte[] GetSession() |
| פונקציה שמקבלת פריים משיתוף המסך שהשרת מעביר ללקוח השולט באמצעות הסוקט מסוג UDP האחראי על שיתוף המסך ומחזיר את ההקלטה כמערך של בתים. | public byte[] GetFrame() |
| פונקציה שמקבלת פריים בפורמט של מערך של בתים ומחזיר את הפריים בפורמט של אובייקט מטיפוס BitmapImage. | public BitmapImage ConvertPngBytesToBitmapImage(byte[] pngBytes) |
| פונקציה שמקבלת פריים בפורמט של BitmapImage ומאחסנת את הפריים בקובץ זמני מסוג PNG. | public string InsertBitmapImageToPngFile(BitmapImage frame) |
| פונקציה שמוציאה את הפריים מקובץ PNG לפי שם הקובץ. | public BitmapImage GetBitmapImageFromPNGFile(string pngFileName) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה ששומר את מספר הפורט שסוקט ה TCP הרגיל משתמש בו לתקשורת בין הלקוח והשרת. | public readonly int ATTACKER\_TCP\_PORT; |
| משתנה ששומר את מספר הפורט שסוקט ה UDP משתמש בו לתקשורת בין הלקוח והשרת. | public readonly int ATTACKER\_UDP\_PORT; |
| משתנה ששומר את מספר הפורט שסוקט ה TCP של ההקלטות משתמש בו לתקשורת בין הלקוח והשרת. | public readonly int ATTACKER\_SESSION\_TCP\_PORT; |
| משתנה מסוג מחרוזת שמאחסן את כתובת ה IP של הלקוח שנשלט. | private string victimIpAddr; |
| אובייקט מסוג BitmapImage (פורמט לתמונות) שאחראי לאחסן את הפריים הנוכחי שהלקוח הנשלט שלך ללקוח השולט דרך השרת. | private BitmapImage currentFrame; |
| משתנה מסוג מערך של בתים ששומר את ההקלטה שהתקבלה מהשרת לאחר בקשה ממסד הנתונים שלו. | private byte[] currentSessionBytes; |
| אובייקט סטטי מטיפוס Attacker שמייצג אובייקט יחידני של מחלקת ה Attacker ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static Attacker attackerInstance = null; |
| משתנה מסוג מחרוזת ששומר את שם הקובץ הזמני של הפריימים של שיתוף המסך בצד הלקוח השולט. | private string tempPngFileName; |

מחלקת **Victim**

מחלקת **Victim** מייצגת קורבן המחובר לשרת מרחוק ומאפשר שליטה בו על ידי תוקף. היא יורשת ממחלקת Client ומתמקדת בטיפול בפקודות מהתוקף - כולל שליחת תמונות מסך, הקלטות, וקבלת פקודות מקלדת ועכבר. המחלקה משתמשת בתבנית העיצוב **Singleton** ומטפלת בחלוקת הנתונים לחבילות לפני שליחתם דרך הרשת. בנוסף, היא מספקת יכולת תגובה לשרת עם סטטוס פעולות כמו אישור קבלת פקודות או עדכון מצב ההקלטה.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. פונקציה זו קוראת ממחלקת האב לפונקציית הבנאי של מחלקת Client. | private Victim() : base() |
| פונקציה ששולחת לשרת בקשת התחברות דרך הסוקטים מסוג TCP של הלקוח הנשלט. | public void ConnectToServer(string serverIpAddr) |
| פונקציה שמריצה באופן א-סינכרוני את הפונקציות שמנהלות את הפרוטוקול של התוכנה בעבור כל סוקט של הלקוח הנשלט. | public void Start() |
| פונקציה שמקבלת מחרוזות של הודעה, סוג הודעה וסוקט ושולחת את ההודעה ללקוח דרך הסוקט אל השרת. | private void SendMessageToClient(string typeOfMessage, string message, Socket socket) |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ושולחת את ההודעה לשרת דרך הסוקט מסוג TCP הרגיל של הלקוח הנשלט. | public void SendTCPMessageToClient(string typeOfMessage, string message) |
| פונקציה שמקבלת מחרוזות של הודעה וטיפוס הודעה ושולחת את ההודעה לשרת דרך הסוקט מסוג UDP של השרת. | public void SendUDPMessageToClient(string typeOfMessage, string message) |
| פונקציה שאחראית להוציא את המידע מההודעה לפי פורמט הפרוטוקול ולהחזירו בצורה מסודרת ונגישה ברשימה. | private List<object> OrderMessage(string message) |
| פונקציה שאחראית לקבל הודעה מסוקט ה TCP הרגיל של השרת, להמיר את ההודעה מטיפוס מערך בתים למחרוזת ולהחזיר את תוכן ההודעה ומאפייניה ברשימה. | public List<object> GetTCPMessageFromClient() |
| פונקציה שאחראית לקבל הודעה מסוקט ה UDP של השרת, להמיר את ההודעה מטיפוס מערך בתים למחרוזת ולהחזיר את תוכן ההודעה ומאפייניה ברשימה. | public List<object> GetUDPMessageFromClient() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהשרת ללקוח הנשלט דרך הסוקט מסוג TCP הרגיל. | private async Task HandleTcpCommunication() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהרשת ללקוח הנשלט דרך הסוקט מסוג UDP. | private async Task HandleUdpCommunication() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהשרת ללקוח הנשלט דרך הסוקט מסוג TCP של ההקלטות. | private async Task HandleSessionsTcpCommunication() |
| פונקציה שמתנהלת עם כל ההודעות שנשלחו מהשרת ללקוח הנשלט דרך הסוקט מסוג TCP של ההקלטות. | private void HandleGeneralStuffMessages(string data) |
| פונקציה שמתנהלת עם הודעות מסוג ניהול משתמשים שהשרת שלח ללקוח הנשלט. | private void HandleUsersManagmentCommands(string data) |
| פונקציה שמתנהלת עם הודעות מסוג שיתוף מסך שהשרת שלח ללקוח הנשלט. | private void HandleFramesCommands(string data) |
| פונקציה שמתנהלת עם הודעות מסוג פקודות עכבר שהשרת שלח ללקוח הנשלט. | private void HandleMouseInput(string message) |
| פונקציה שמתנהלת עם הודעות מסוג פקודות מקלדת שהשרת שלח ללקוח הנשלט. | private void HandleKeyboardInput(string message) |
| פונקציה שמתנהלת עם הודעות מסוג ניהול הקלטות שהשרת שלח ללקוח הנשלט. | private void HandleSessionsCommands(string message) |
| פונקציה שמקבלת הקלטה מהשרת בסוקט מסוג TCP האחראי על ההקלטות ומחזיר את ההקלטה כמערך של בתים. | private byte[] GetSession() |
| פונקציה שמאפסת את המשתנה **serverRespond** באמצעות מחרוזת ריקה. | public void InitServerRespond() |
| פונקציה שיוצרת קובץ PNG זמני ומכניסה אליו את הפריים הנוכחי משיתוף המסך. | public void CreatePngFile(RenderTargetBitmap frame, string newFileName) |
| פונקציה שפותחת קובץ ומחזירה את התוכן הבינארי שלו כמערך בינארי. | public byte[] GetFileContent(string fileName) |
| פונקציה ששולחת פריים לשרת (והשרת מייעד אותו ללקוח השולט). | public void SendFrame(byte[] frame) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה ששומר את מספר הפורט שסוקט ה TCP הרגיל משתמש בו לתקשורת בין הלקוח והשרת. | public readonly int VICTIM\_TCP\_PORT; |
| משתנה ששומר את מספר הפורט שסוקט ה UDP משתמש בו לתקשורת בין הלקוח והשרת. | public readonly int VICTIM\_UDP\_POR; |
| משתנה ששומר את מספר הפורט שסוקט ה TCP של ההקלטות משתמש בו לתקשורת בין הלקוח והשרת. | public readonly int VICTIM\_SESSION\_TCP\_PORT |
| משתנה ששומר את אורך המסך (שמתקבל אוטומטית מה WINAPI). | private int screenWidth; |
| משתנה ששומר את רוחב המסך (שמתקבל אוטומטית מה WINAPI). | private int screenHeight; |
| משתנה מטיפוס מחרוזת שמאחסן את כתובת ה IP של הלקוח השולט. | private string attackerIpAddr; |
| משתנה מסוג מערך של בתים ששומר את ההקלטה שהתקבלה מהשרת לאחר בקשה ממסד הנתונים שלו. | private byte[] currentSessionBytes; |
| אובייקט סטטי מטיפוס Victim שמייצג אובייקט יחידני של מחלקת ה Victim ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static Victim victimInstance; |

מחלקת **User**

מחלקת **User** מייצגת את המשתמש במערכת, בין אם הוא תוקף או קורבן. היא כוללת את שם המשתמש, סוג הלקוח (תוקף/קורבן) ורשימת ההקלטות שלו. המחלקה מאפשרת גישה ועדכון לפרטי המשתמש בצורה מבוקרת.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | public User() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה מסוג מחרוזת ששומר את שם המשתמש של הלוקח. | private string username; |
| משתנה מסוג TypeOfClient (Enum) שקובע את סוג הלקוח. | private TypeOfClient typeOfClient; |
| משתנה מסוג מערך IList ששומר אובייקטים של הקלטות. | private IList<Session> userSessions; |
| אובייקט סטטי מטיפוס User שמייצג אובייקט יחידני של מחלקת ה User ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static User userInstance; |

מחלקת **Session**

מחלקת Session מייצגת הקלטת מסך (session) במערכת. היא כוללת את תאריך ההקלטה, תוכן הווידאו (כמערך בתים) ומזהה ייחודי. המחלקה מספקת פונקציונליות לשליחת בקשות לשרת - הן לקבלת רשימת ההקלטות של משתמש מסוים והן לקבלת תוכן הקלטה ספציפית לפי ID.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | public Session() |
| פונקציית הבנאי של המחלקה שמקבלת את זמן ההקלטה של ההקלטה ואת המספר המזהה של המחלקה. | public Session(DateTime recordedTime, int id) |
| פונקציה שיוצרת בקשה בפורמט JSON בעבור השרת על מנת לקבל את מאפייני ההקלטות של כל ההקלטות של המשתמש של הלקוח. | private static string GenerateGetAllSessionsPropertiesQuery(TypeOfSessionRequest requestType, string username) |
| פונקציה שיוצרת בקשה בפורמט JSON בעבור השרת על מנת לקבל את התוכן של הקלטה ספציפית של המשתמש של הלקוח לפי המספר המזהה של ההקלטה. | private static string GenerateGetSessionContent(TypeOfSessionRequest requestType, int id) |
| פונקציה ששולחת את הבקשה הנבחרת לשרת. | private static void SendSessionRequestToServer(string message) |
| פונקציה שיוצרת ושולחת את הבקשה לפי סוג הבקשה שהתקבל. | public static void SendRequest(TypeOfSessionRequest requestType, int id=0) |
| פונקציה שמקבלת מחרוזת שמכילה הודעה בפורמט json שמכילה תכונות של הקלטות, ממירה אותה לאובייקט של מערך IList של הקלטות ומחזירה אותו. | public static IList<Session> FromSessionPropertiesListJsonStrToSessionIlist(string sessionPropertiesList) |
| פונקציה שיוצרת קובץ MP4 (פורמט לקבצי ווידאו) שמכיל את ההקלטה שהתקבלה מהשרת. | public static void CreateTempMP4File(byte[] videoContent) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה שמכיל את זמן ההקלטה של ההקלטה בפורמט של אובייקט DateTime (אובייקט שאחראי לאחסן זמנים ותאריכים). | private DateTime recordedTime; |
| משתנה מסוג מערך של בתים ששומר את תוכן ההקלטה. | private byte[] videoContent; |
| משתנה מסוג מספר שלם שמכיל את המספר המזהה של ההקלטה. | private int id; |

מחלקת **AttackerSession**

מחלקת **AttackerSession** אחראית על הקלטת וניהול סרטוני המסך מהצד התוקף. היא מספקת יכולת להתחיל ולהפסיק הקלטות, להוסיף פריימים לסרטון בזמן אמת, ולשלוח את ההקלטות לשרת.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | public AttackerSession(int width, int height) |
| פונקציה שמאתחלת את הסשיין ומאפשרת לו להתחיל לקבל פריימים. | public void Start() |
| פונקציה שממירה פריים מטיפוס BitmapImage לטיפוס Bitmap. | private Bitmap BitmapImageToBitmap(BitmapImage bitmapImage) |
| פונקציה שמקבלת פריים ומוסיפה אותו להקלטה. | public void WriteFrame(BitmapImage bitmapImage) |
| פונקציה שממירה את הקובץ MP4 הזמני של ההקלטה. | private byte[] ConvertMP4ToByteArr() |
| פונקציה שסוגרת את ההקלטה ולא מאפשרת להקלטה לקבל עוד פריימים. | public void Stop() |
| פונקציה ששולחת את תוכן ההקלטה לשרת כדי שהשרת יאחסן אותו במסד הנתונים של ההקלטות. | private void SendSessionContentToServer() |
| פונקציה ששולחת את מאפייני ההקלטה לשרת כדי שהשרת יאחסן אותם במסד הנתונים של ההקלטות. | public void SendSessionToServer() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| קבוע מטיפוס מספר שלם ששומר את כמות הפריימים בשנייה שיש בכל הקלטה. | private readonly int FPS; |
| משתנה ששומר את זמן ההקלטה של ההקלטה. | private DateTime recordedTime; |
| משתנה מסוג מערך של בתים ששומר את תוכן ההקלטה. | private byte[] videoContent; |
| משתנה מסוג VideoWriter (אובייקט שמשמש ליצירת ווידאו באמצעות פריימים) ששומר את הפריימים עצמם לפי הסדר של ההקלטה. | private VideoWriter videoWriter; |
| משתנה מסוג מחרוזת ששומר את שם הקובץ הזמני של ההקלטה. | private string tempVideoPath; |
| משתנה מסוג מספר שלם ששומר את אורך (בפיקסלים) של פריים אחד בהקלטה. | private int width; |
| משתנה מסוג מספר שלם ששומר את הרוחב (בפיקסלים) של פריים אחד בהקלטה. | private int height; |
| משתנה שמהווה Mutex Lock(אובייקט שמונע התנגשות בין 2 פונקציות שמשתמשות באותם משאבים כמו לדוגמה משתנים) שאחראי לוודא שפונקציית סגירת ההקלטה לא מתנגשת עם פונקציית הוספת פריים להקלטה בו זמנית. | private readonly object lockObj; |
| משתנה בוליאני שמעיד על האם ההקלטה פעילה או לא פעילה. | private bool isActive; |

מחלקת **CustomMouseAttacker**

מחלקת **CustomMouseAttacker** מייצגת את ממשק העכבר של התוקף ומאפשרת שליטה מרחוק בעכבר הקורבן. היא בנויה לפי תבנית העיצוב **Singleton** ומנהלת את מיקום הסמן ופעולות העכבר. המחלקה יוצרת פקודות JSON הכוללות את סוג הפעולה (תזוזה/לחיצה) ומיקום הסמן, ושולחת אותן דרך השרת לקורבן.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | private CustomMouseAttacker() |
| פונקציה שמשנה את המיקום של המשתנה **cursorsPos** למיקום החדש שהלקוח השולט בחר לפי אובייקט ה point שניתן לפונקציה (אובייקט שמיועד לשמירה של ערכי מיקום). | public void ChangePosition(Point newPos) |
| פונקציה שמשנה את המיקום של המשתנה **cursorsPos** למיקום החדש שהלקוח השולט בחר לפי הפרמטרים x ו y שניתנו לפונקציה. | public void ChangePosition(int newXPos, int newYPos) |
| פונקציה שיוצרת הודעה בפורמט json בעבור הלקוח הנשלט שעליו לשנות את מיקום העכבר שלו לפי הערכים החדשים. | private string GeneratePositionQuery() |
| פונקציה שיוצרת הודעה בפורמט json בעבור הלקוח הנשלט שעליו ללחוץ על הכפתור השמאלי של העכבר. | private string GenerateLeftPressingQuery() |
| פונקציה שיוצרת הודעה בפורמט json בעבור הלקוח הנשלט שעליו ללחוץ על הכפתור ימני של העכבר. | private string GenerateRightPressingQuery() |
| פונקציה ששולחת את ההודעה לשרת (שישלח אותה ללקוח הנשלט) שניתנה לה כהודעת פקודת עכבר. | private void SendCommandQueryToClient(string commandQuery) |
| פונקציה שמקבלת את סוג הפקודה שעליה ליצור, יוצרת אותה ושולחת אותה. | public void SendMouseCommands(AttackerMouseCommands mouseCommand) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| אובייקט מסוג Point (אובייקט לשמירת מיקום במישור) ששומר את הערך של מיקום העכבר שהלקוח השולט שולח לנשלט. | private Point cursorsPos; |
| אובייקט סטטי מטיפוס CustomMouseAttacker שמייצג אובייקט יחידני של מחלקת ה CustomMouseAttacker ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static CustomMouseAttacker customMouseInstance; |

מחלקת **CustomMouseVictim**

מחלקת **CustomMouseVictim** אחראית על ביצוע פעולות העכבר בפועל במחשב הקורבן. היא מקבלת פקודות מהתוקף (כמו תזוזה או לחיצות) ומתרגמת אותן לפעולות ממשיות על מערכת ההפעלה של הקורבן באמצעות קריאות ל **WIN API.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | private CustomMouseAttacker() |
| פונקציה שמשנה את המיקום של המשתנה **cursorsPos** למיקום החדש שהלקוח השולט בחר לפי אובייקט ה point שניתן לפונקציה (אובייקט שמיועד לשמירה של ערכי מיקום). | public void ChangePosition(Point newPos) |
| פונקציה שמשנה את המיקום של המשתנה **cursorsPos** למיקום החדש שהלקוח השולט בחר לפי הפרמטרים x ו y שניתנו לפונקציה. | public void ChangePosition(float newXPos, float newYPos) |
| פונקציה שמשנה את מיקום הסמן של העכבר במחשב של הלקוח הנתקף לפי הערכים של המשתנה **cursorsPos.** | private void MoveMouseToPosition() |
| פונקציה שמדמה לחיצה על הכפתור השמאלי של העכבר במחשב של הלקוח הנתקף. | private void LeftClick() |
| פונקציה שמדמה לחיצה על הכפתור השמאלי של העכבר במחשב של הלקוח הנתקף. | private void RightClick() |
| פונקציה שמבצעת את פקודת העכבר שהלקוח השולט שלח ללקוח הנשלט. | public void ExecuteCurrentMouseCommand() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה מסוג Point ששומר את המיקום החדש של הסמן של העכבר של הלקוח הנשלט. | private Point cursorsPos; |
| משתנה מסוג VictimMouseCommands (Enum) שקובע את סוג הפקודה האחרונה שניתנה ללקוח הנשלט. | private VictimMouseCommands currentCommand; |
| אובייקט סטטי מטיפוס CustomMouseVictim שמייצג אובייקט יחידני של מחלקת ה CustomMouseVictim ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static CustomMouseVictim customMouseInstance; |

מחלקת **CustomKeyboardAttacker**

מחלקת **CustomKeyboardAttacker** מייצגת את ממשק המקלדת בצד התוקף ומאפשרת שליטה מרחוק במקלדת הקורבן. המחלקה בנויה על פי תבנית העיצוב **Singleton.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | private CustomKeyboardAttacker() |
| פוקנציה שיוצרת הודעה בפורמט json בעבור הלקוח הנשלט שעליו ללחוץ על כפתור מסוים במקלדת. | private string GenerateKeyPressQuery() |
| פונקציה ששולחת את הודעת פקודת מקלדת ללקוח (דרך השרת) | private void SendCommandQueryToClient(string commandQuery) |
| פונקציה שמקבלת כפתור במקלדת (משתנה מסוג Key (Enum)), יוצרת הודעת פקודת מקלדת ושולחת אותה. | public void SendKeyboardCommands(Key key) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה ששומר את הכפתור הנוכחי שהלקוח השולט לחץ במקלדת. | private Key currentKey; |
| משתנה מסוג AttackerKeyboardCommands (Enum) שקובע את סוג הפקודה האחרונה שניתנה ללקוח הנשלט. | private AttackerKeyboardCommands currentKeyboardCommand; |
| משתנה מסוג Action מסוג Key שמתעדכן כאשר הלקוח לוחץ על כפתור כלשהו. | public event Action<Key> KeyPressed; |
| משתנה מסוג Action מסוג Key שמתעדכן כאשר הלקוח עוזב כפתור כלשהו שהוא לחץ עליו. | public event Action<Key> KeyReleased; |
| משתנה מסוג LowLevelKeyboardProc שאחראי לאחסן את ה delegate(אובייקט שמהווה Pointer לפונקציות) של הפונקציה LowLevelKeyboardProc על מנת לשמור את הערכים שלה כדי לקבל גישה אליהם מה WIN API. | private LowLevelKeyboardProc proc; |
| משתנה מסוג IntPtr שאחרי לשמור את המספר המזהה של Windows keyboard hook (מערכת low level שמאזינה לקלטי מקלדת). | private IntPtr hookID; |

מחלקת **CustomKeyboardVictim**

מחלקת **CustomKeyboardVictim** מייצגת את ממשק המקלדת בצד הקורבן ומבצעת בפועל את פקודות המקלדת שמתקבלות מהתוקף. המחלקה בנויה על פי תבנית העיצוב **Singleton.**

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה | private CustomKeyboardVictim() |
| פונקציה שמדמה לחיצה על כפתור מסוים במקלדת. | public void PressKey(Key keyToPress) |
| פונקציה שמבצעת את פקודת המקלדת שניתנה מהלקוח השולט. | public void ExecuteCurrentKeyboardCommand() |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| משתנה מסוג VictimKeyboardCommands (Enum) שמאחסן את סוג פקודת המקלדת שניתנה. | private VictimKeyboardCommands currentKeyboardCommand; |
| משתנה מסוג Key (Enum) ששומר את הכפתור שעליו יש ללחוץ במקלדת. | private Key currentKey; |
| אובייקט סטטי מטיפוס CustomKeyboardVictim שמייצג אובייקט יחידני של מחלקת ה CustomKeyboardVictim ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static CustomKeyboardVictim customKeyboardInstance = null; |

מחלקת **SoundManager**

מחלקת **SoundManager** אחראית על ניהול ושליטה בקבצי האודיו בתוכנה. היא בנויה על פי תבנית העיצוב **Singleton** ומספקת ממשק פשוט להפעלת מוזיקה בתוכנה.

|  |  |
| --- | --- |
| **תפקיד הפונקציה** | **פונקציה** |
| פונקציית הבנאי של המחלקה. | private SoundManager() |
| פונקציה שמקבלת את שם השיר (מרשימת הפלייליסט שמיוצגת באמצעות ה Enum של EPlaylist) ומחזירה את הקובץ שמע של הסאונד לפי הנתיב של הקובץ שהוגדר מראש. | private string GetSongPath(EPlaylist song) |
| פונקציה שאחראית לנגן את הסאונד מהקובץ הנבחר. | public void PlaySound(EPlaylist sound) |

|  |  |
| --- | --- |
| **תפקידם** | **התכונות של המחלקה** |
| אובייקט מסוג WaveOutEvent שאחראי להשמיע את הקול של הסאונדים עם השהיה נמוכה (משמיע את הצליל בפועל). | private WaveOutEvent outputDevice; |
| משתנה מסוג AudioFileReader (אובייקט שאחראי לקרוא תוכן של קבצים) שאחראי לקרוא את התוכן של קבצי השמע של סאונדים. | private AudioFileReader audioFile; |
| אובייקט סטטי מטיפוס SoundManager שמייצג אובייקט יחידני של מחלקת ה SoundManager ובכך מיישם את תבנית העיצוב Singleton במחלקה זו. | private static SoundManager soundManagerInstance; |

רפלקציה

**במהלך העבודה על פרויקט** ExtremLink**, עברתי תהליך אישי ומקצועי משמעותי. מדובר לא רק בפרויקט גמר טכנולוגי, אלא בהתנסות מעשית ראשונה שלי בבנייה של מערכת מורכבת מהיסוד – החל משלב הרעיון, דרך מחקר תיאורטי ויישום טכנולוגי, ועד מוצר גמור ומתפקד.**

**העבודה על הפרויקט דרשה ממני התמודדות עם אתגרים טכנולוגיים לא פשוטים, בעיקר סביב נושא תקשורת בזמן אמת והעברת פריימים בין מחשבים שונים. היה עליי להבין לעומק את עקרונות מודל השרת-לקוח, לבחון את היתרונות והחסרונות של פרוטוקולי תקשורת כמו** TCP **ו-**UDP**, ולהתנסות בשיטות המרה ודחיסה של נתונים ויזואליים. לא הסתפקתי רק בתיאוריה – חיפשתי דוגמאות קוד, שאלתי מומחים בפורומים, ובחנתי פתרונות שונים עד שמצאתי את הדרך האופטימלית עבורי.**

**ההחלטה להשתמש בפרוטוקול** UDP **למשל, לא הייתה מובנת מאליה – אבל היא התבררה כהכרחית כדי לאפשר שידור רציף ומהיר של תוכן וידאו חי, גם אם במחיר של אמינות חלקית. באותה מידה, ההתעסקות עם קבצים זמניים, המרות לפורמטים בינאריים והקלטות סשנים הייתה מאתגרת אך גם מהנה. כל פעם שפתרתי בעיה – הרגשתי איך אני מתקדם ברמה האישית, לא פחות מאשר ברמה הטכנולוגית.**

**לאורך הדרך, למדתי הרבה מעבר לתכנות בשפת** C# **או עבודה עם מסדי נתונים. למדתי לתכנן מערכת מקצה לקצה, להבין את הצרכים של המשתמש, לחשוב על אבטחת מידע, לבצע תיעוד מסודר, ולפתח ממשק משתמש נעים וברור באמצעות** WPF**. התקשורת בין רכיבי המערכת, שמירת ההיסטוריה של הסשנים, והאפשרות לצפות בהם לאחר מכן – כל אלו דרשו ממני תכנון מדויק, כתיבה של מחלקות רבות והתמדה אמיתית.**

**אחת התובנות הכי חשובות שיצאתי איתן מהפרויקט היא שאני מסוגל ליצור משהו משמעותי מאפס. זה חיזק בי את התחושה שאני לא רק לומד תכנות – אני באמת הופך למפתח תוכנה. מעבר לכך, אני מאמין שהפרויקט ישמש אותי גם בהמשך הדרך – בין אם כבסיס לפרויקטים מתקדמים יותר ובין אם כתוספת משמעותית לתיק העבודות שלי לקראת השתלבות בשוק ההיי-טק.**

**לסיכום, פרויקט** ExtremLink **היה עבורי אתגר גדול – אבל גם הזדמנות נדירה להוכיח לעצמי שאני מסוגל להתמודד עם משימות מורכבות, ללמוד לבד נושאים מתקדמים, ולספק פתרון אמיתי לצורך ממשי. אני גאה בתוצאה, ובעיקר – בדרך שעברתי עד אליה.**

ביבליוגרפיה

**<https://www.codingwithsaar.co.il/articles/client-server-model>**

**<https://shushan.co.il/%D7%94%D7%A1%D7%91%D7%A8-%D7%A2%D7%9C-tcp-%D7%95-udp>**

**<https://webmaster.org.il/wpf/>**

**<https://visualstudio.microsoft.com/vs/features/ssdt/>**

**<https://reshetech.co.il/javascript-tutorials/what-is-json>**

**<https://webmaster.org.il/xml-first-steps/>**

**<https://codetime.co.il/singleton/>**

**<https://webmaster.org.il/csharp-oop-abstract/>**

**<https://learn.microsoft.com/en-us/dotnet/api/system.collections.ilist?view=net-9.0>**