שם התלמיד: יובל כוינה שם הפרויקט: ExtremLink

**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:

מבוא:

תיאור תכולת הספר:

ספר הפרויקט שלי כולל מבוא, אשר מתאר את מטרת הפרויקט, תהליך המחקר ואתגרים מרכזיים שאיתם התמודדתי. בנוסף לכך, הספר כולל גם ארכיטקטורה של הפרויקט, מדריך למשתמש, הסבר על בסיסי הנתונים שיצרתי ומדריך למפתח, כאשר על פרק יסביר יותר לעומק על הפרויקט שלי ומרכיביו. בסוף הספר מופיעים 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.

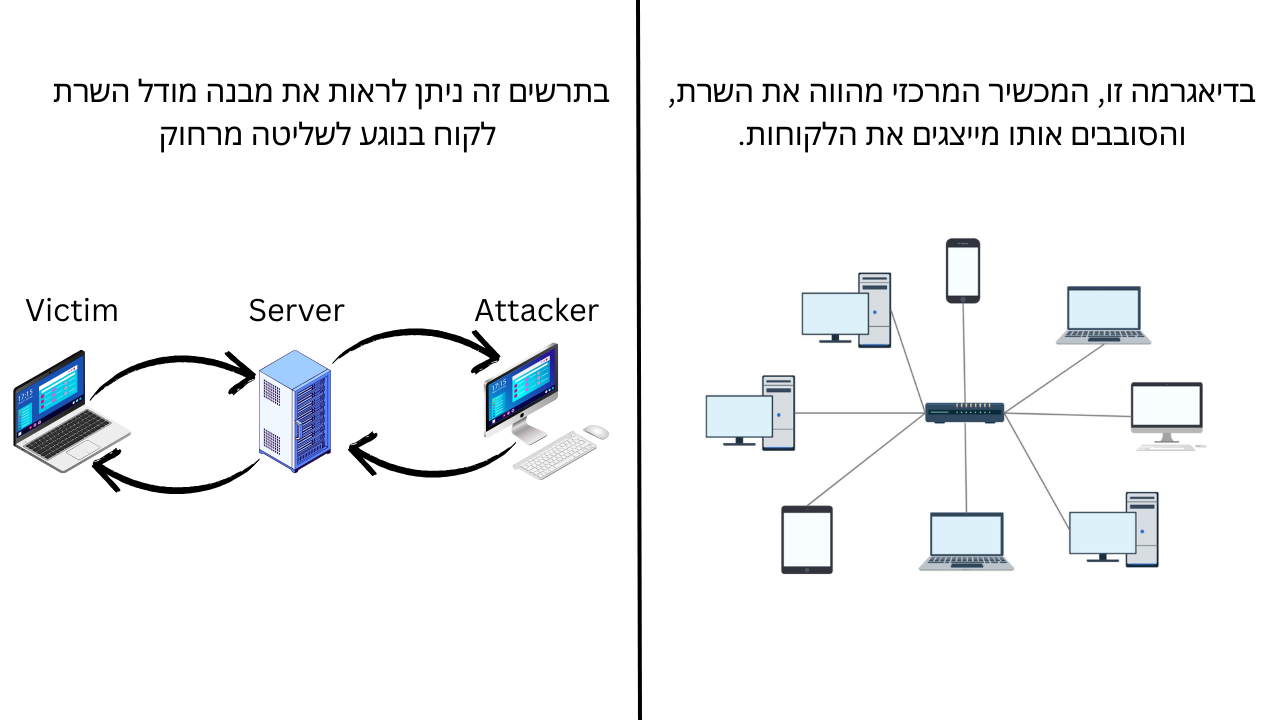
הפרויקט מאפשר תקשורת בין המשתמשים אשר הוזנו במערכת, כאשר התקשורת נעשית על ידי שיתוף מסך המחשב של הלקוח המשתף אל הלקוח השולט. לשם הרשמה למערכת ויצירת חשבון חדש, על הלקוח להזין שם משתמש, סיסמה, את שמו הפרטי, את שם המשפחה שלו, את מקום מגוריו, מספר הטלפון שלו ואת כתובת המייל שלו. לאחר שנתוני הלקוח הוזנו, 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:

תהליך המחקר המקדים לפרויקט שאותו ביצעתי, מהווה חקירה לעומק על מודל שרת-לקוח. מודל זה הינו ארכיטקטורת תוכנה לחישוב מבוזר, אשר מגדירה את היחס בין תוכנות משתפות פעולה על בסיס רשת דיגיטלית. כלומר, המודל מורכב משני חלקים, כאשר הוא מחלק את המשימות או את עומס העבודה ביניהם. יישום השרת הינו אחד מהחלקים המרכיבים את המודל, ותפקידו לספק את השירותים/משאבים אל מבקשי השירות. אלו נקראים הלקוחות אשר מהווים את החלק השני של המודל.

**השרת** הינו תוכנה פסיבית המאזינה לרשת באופן תמידי, ומחכה לקבל בקשות. לעומת זאת, **הלקוח**, שהוא לרוב מהווה את ממשק המשתמש, מופעל על ידי המשתמש והוא פונה לשרת כאשר הוא זקוק למידע או לשירותים ממנו. אופן התקשורת בין שני חלקי היישום נעשית על ידי העברת הודעות כפי שנקבע בפרוטוקול התקשורת. כלומר, כאשר לקוח מעוניין לקבל שירות מהשרת, הוא שולח לשרת בקשה ליצירת קשר. לאחר מכן, השרת אשר מאזין ומחכה לקבל בקשות, מקבל את הבקשה ומשיב עליה. מכאן, שניתן להגדיר את הלקוח כיוזם קשר, ואת השרת כמאזין ליצירת קשר. הלקוח בדרך כלל נדרש לדעת את שמו של השירות אותו הוא מעוניין לקבל, ובנוסף את שמו או שמותיהם של השרת או השרתים הזמינים על מנת להגיע אליהם. בניגוד לכך, השרת אינו צריך לדעת מי הם לקוחותיו ואת מספר הלקוחות שיש.

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.

ישנם שני צורות שלפיהם מודל שרת לקוח יכול לפעול, כאשר בדרך כלל תוכנות השרת והלקוח רצות על גבי מחשבים שונים והתקשורת ביניהם מתבצעת על גבי רשת מחשבים. אולם, ישנם מקרים שבהם המודל פועל לפי צורת הפעולה השנייה, כאשר היא מתבססת על כך שהתוכנות רצות על גבי אותו המחשב. מודל השרת לקוח הפך לאחד מהרעיונות המרכזיים ברשתות מחשבים, כאשר מספר רב של יישומים אשר נכתבים היום משתמשים ומבוססים על מודל זה. שירותים כגון: גישה לאתרי אינטרנט, גישה לבסיסי נתונים, וכמו כן גם הפרויקט שלי אשר מהווה שירות המאפשר תקשורת בין מחשבים תוך שימוש בשרת המיועד לקבל ולשלוח נתונים ללקוחות המחוברים אליו כמו לדוגמה בעת ההשתלטות שהלקוח המשתף שולח את הפריימים מהמסך שלו אל השרת והשרת בתגובה מעביר אותם למחשב 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. תמונה שמכילה שרטוט, תרשים, עיצוב

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

היתרונות של הרשת כוללים את זה שרשת כזו הינה קלה לתחזוקה, וניתן להסיר ולהוסיף אליה קצוות בקלות. בנוסף, אם אחד מן המכשירים ההיקפיים הנמצאים ברשת מפסיק לפעול, זה אינו משפיע על המכשירים האחרים והרשת ממשיכה לפעול כרגיל.

עם זאת, לטופולוגית כוכב ישנם חסרונות הכוללים את זה שהיא בזבזנית במקצת באורכי הכבלים שהיא דורשת. בנוסף, הצומת המרכזי מהווה נקודת חולשה בטופולוגית כוכב, מכיוון שבמידה וצומת זו אינו פעיל, כל הרשת אינה יכולה לתפקד. יתר על כן, מכשיר הרשת המרכזי קובע את הביצועים ואת מספר הצמתים שהרשת יכולה להתמודד איתם.

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!

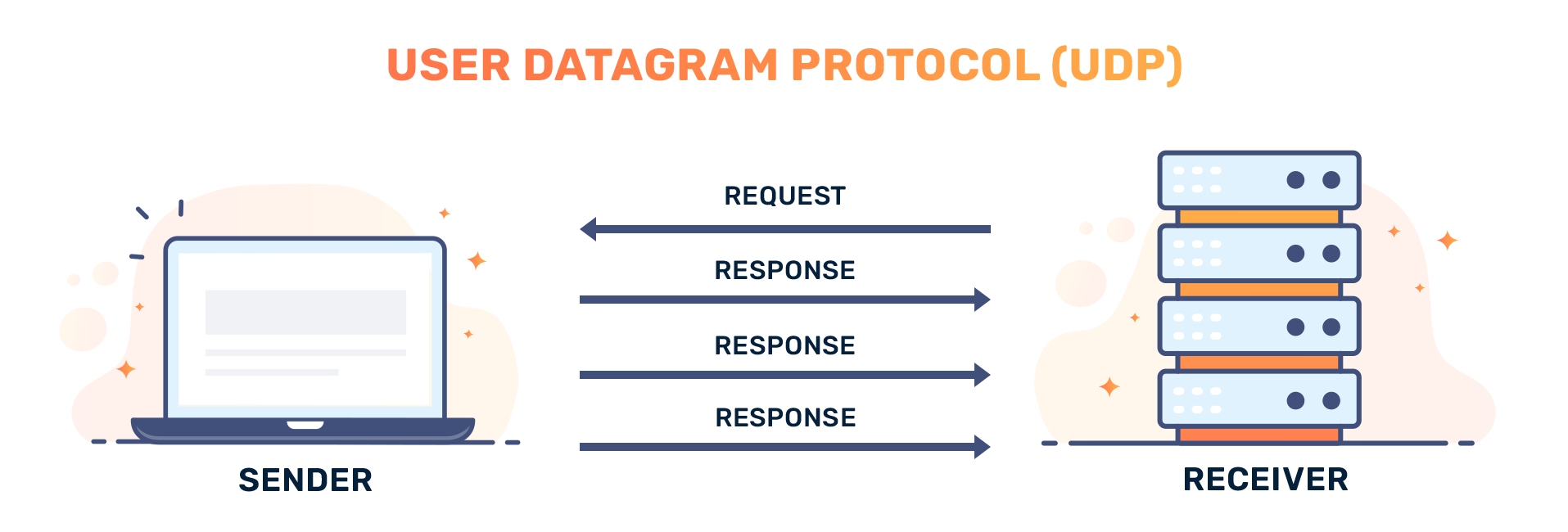
הצורך שעליו עונה הפרויקט -

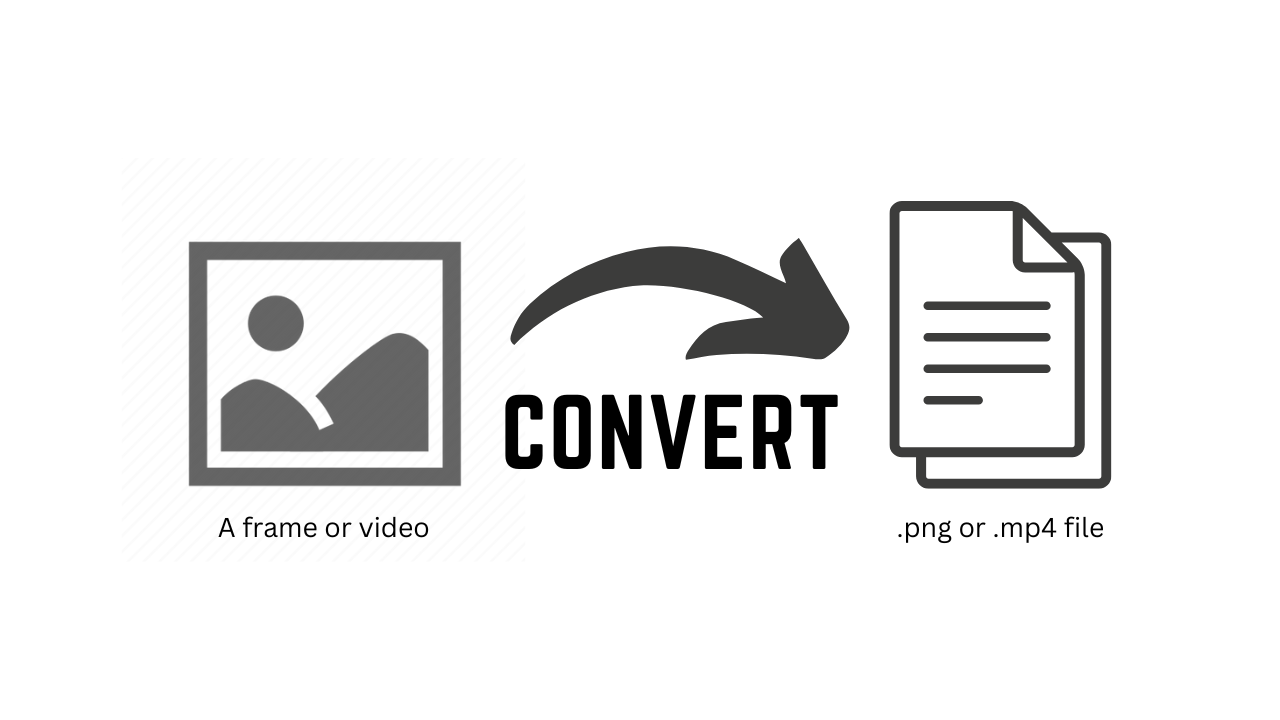
הפרויקט שלי עונה על הצורך לאמצעי תקשורת זמין ומהיר, כאשר לקוח מסוים נתקל בבעיה כלשהי במחשבו האישי וברצונו שמפעיל מרחוק כגון טכנאי או כמגיש שירות לקוחות יתקן את התקלה מרחוק ללא צורך בהגעה פיזית. פתרון זה יפתח צוהר לשירותים חדשים בעולמות טכנאות המחשבים ויקל על הלקוחות.

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 **Victim**לא 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 **User**לא 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 |
| המחלקה**CustomKeyboardVictim** 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 הינה רשת המכילה כמות עצומה של אתרי אינטרנט, אשר נגישים לציבור הכלל עולמי. לכל אתר אינטרנט הנמצא ברשת יש דומיין (לדוגמה: “google.com”), מעין מתחם בו הוא מתארח, כאשר לכל דומיין יש שם (domain name). שם זה הינו ייחודי לכל אתר אינטרנט, ובאמצעותו ניתן להבדיל בין כל האתרים הקיימים ברשת. שם המתחם המקורי מורכב מספרות עם נקודות המבדילות ביניהם, כלומר הוא כתובת IP. היות ומחשבים עובדים על בסיס של מספרים ובני אדם לפי אותיות, יש צורך בפרוטוקול DNS (Domain Name System).

DNS הינו פרוטוקול המאפשר גישה לבסיס נתונים מבוזר, כאשר שרתי הDNS אחראים על ההמרה של שמות המתחם מכתובת IP לכתובת URL המהווה כתובת מילולי. כך מתאפשרת גישה של בני אדם אל אתרי האינטרנט בצורה פשוטה יותר. באמצעות הDNS ניתן להציע שירותים מבוססי שם נוספים, כגון: רישום של שרתי דואר. כמו כן, הפרויקט שלי הינו דואר אלקטרוני בעל בסיס נתונים מבוזר, המשתמש בפרוטוקול DNS. בנוסף, שרתי DNS יכולים לתת שירות למספר רב של דומיינים. כל דומיין מקבל תחום (Zone), כאשר כל תחום הוא למעשה טבלה, הכוללת מספר סוגי רשומות ולכל רשומה שדות רלוונטיים. על מנת לבצע העברות של תחומים (Zone Transfer) בין שרת DNS ראשי לבין שרת DNS משני, מתקיימת שימוש בפרוטוקול התקשורת TCP דרך פורט 53, על ידי שרת הDNS.

מודל שרת לקוח עושה שימוש בחבילת פרוטוקולי התקשורת IP/TCP, כאשר הוא מורכב מפרוטוקול התקשורת TCP ופרוטוקול התקשורת IP:

פרוטוקול (Transition Control Protocol) TCP הינו פרוטוקול אשר מבצע בדיקות בזמן העברת המידע בין שתי נקודות שונות. המידע מחולקת לחבילות קטנות, כאשר מוגדרת לכל חבילה מספר סידורי. תפקידו של הפרוטוקול הינו לסדר את החבילות אשר מגיעות למקור היעד ולוודא שהם בסדר סידורי רץ. יש צורך בבדיקה זו מכיוון שהחבילות לא מגיעות באותו הסדר שבו הן נשלחו, וזה עקב ניתובים שונים. במידה וחסרות חבילות מידע במקור היעד, פרוטוקול TCP שולח את החבילות החסרות עוד פעם ומציג את המידע רק כאשר הוא שלם ובלי חלקים חסרים. פרוטוקול TCP הינו הפרוטוקול הנפוץ והעיקרי ברשת, ויישומים אשר משתמשים בו הינם יישומים ששלמות המידע חשובה להם יותר מאשר המהירות שבה המידע עוברת מנקודה לנקודה.

פרוטוקול IP (Internet Protocol) הינו פרוטוקול תקשורת המשמש להעברת נתונים, כאשר הוא לא כולל בדיקת הגעה או בדיקת הנתונים. הוא מחלק את המידע שעליו להעביר לחבילות, כאשר כל חבילה מורכבת מפתיח ונתונים ואינה מכילה סוגר בניגוד לפרוטוקולים אחרים. בנוסף, הפרוטוקול מטפל בכתובת אליה נשלחת החבילה כדי שיגיע ליעד הנכון. הפרוטוקול הינו יעיל ומהיר ביותר, ולכן הוא מהווה פרוטוקול הנפוץ בשימושו ברשתות מחשבים. בפרוטוקול הIP כל מחשב ברשת מקבל כתובת IP ייחודית משלו, וכך ניתן להבדיל בין המחשבים ולשלוח מידע ברשת למחשב הרצוי.

IP/TCP היא חבילת פרוטוקולי תקשורת שעליה מבוססת רשת האינטרנט, והיא בעלת שתי שכבות: השכבה העליונה הינה פרוטוקול הTCP, והשכבה התחתונה הינה פרוטוקול הIP. כמו כן, מודל שרת לקוח עושה שימוש בחבילה זו לשם תקשורת בין השרת לבין הלקוחות, כך שמחשב הלקוח יוכל לבקש ולקבל שירות ממחשב השרת ברשת. תקשורת IP/TCP הינה תקשורת המתנהלת בין נקודה לנקודה, ובמסגרת התקשורת כל בקשה של לקוח מהווה בקשה חדשה ונפרדת שאינה תלויה בבקשה הקודמת לה. היתרון בכך הוא שתקשורת זו מאפשרת שחרור של נתיבי התקשורת, כך שכולם יכולים להשתמש בהם באופן רציף.

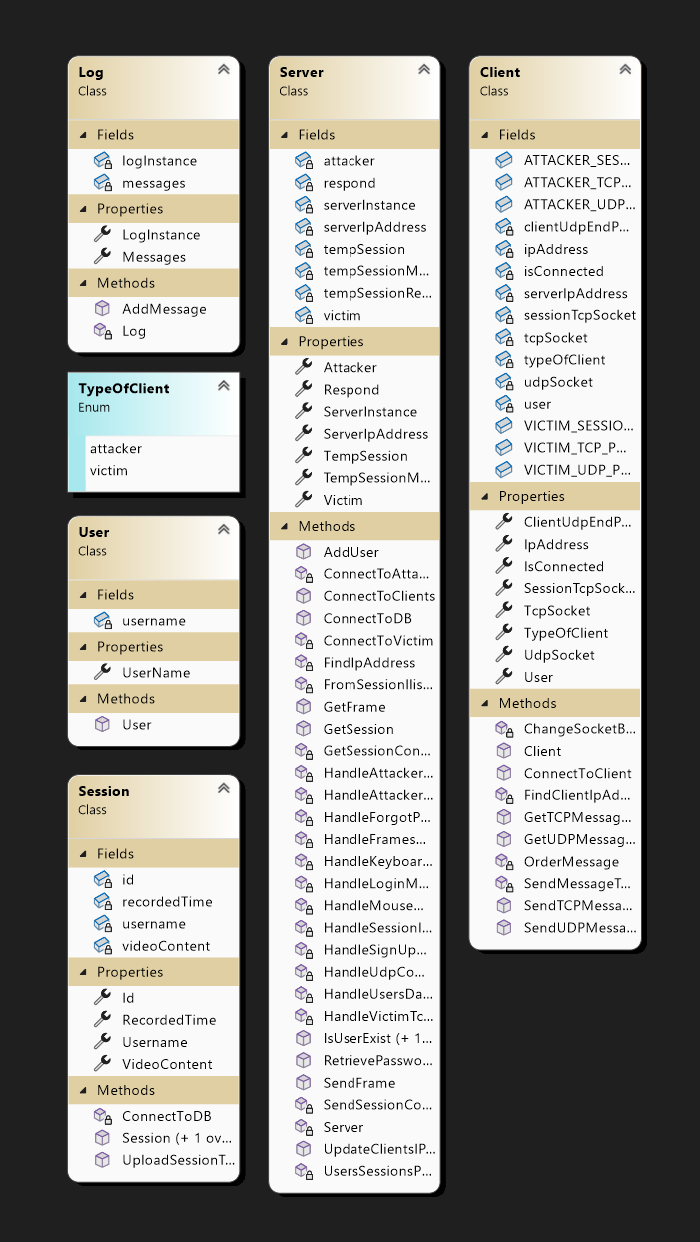
בנוסף המודל יכול לכלול שימוש בפרוטוקולים אחרים לפי מודל ה OSI כמו פרוטוקול ה UDP(User Datagram Protocol) שנעשה בו שימוש בפרויקט שלי. פרויקט זה שנמצא בשכבת התעבורה (כמו פרוטוקול TCP) אחראי להעברת מידע בין 2 נקודות קצה. ההבדל המרכזי בין TCP ל UDP הוא ש UDPנחשב למהיר יותר לעומת TCP מכיוון שהוא אינו מבצע בדיקת אמינות של המידע ששהתקבל, כך שהמידע יוכל להגיע באופן חלקי לאור איבוד של חבילות מידע במהלך התקשורת ללא השלמתן כמו בפרוטוקול TCP.

**הטכנולוגיות שבהן נעשה שימוש בפרויקט:**

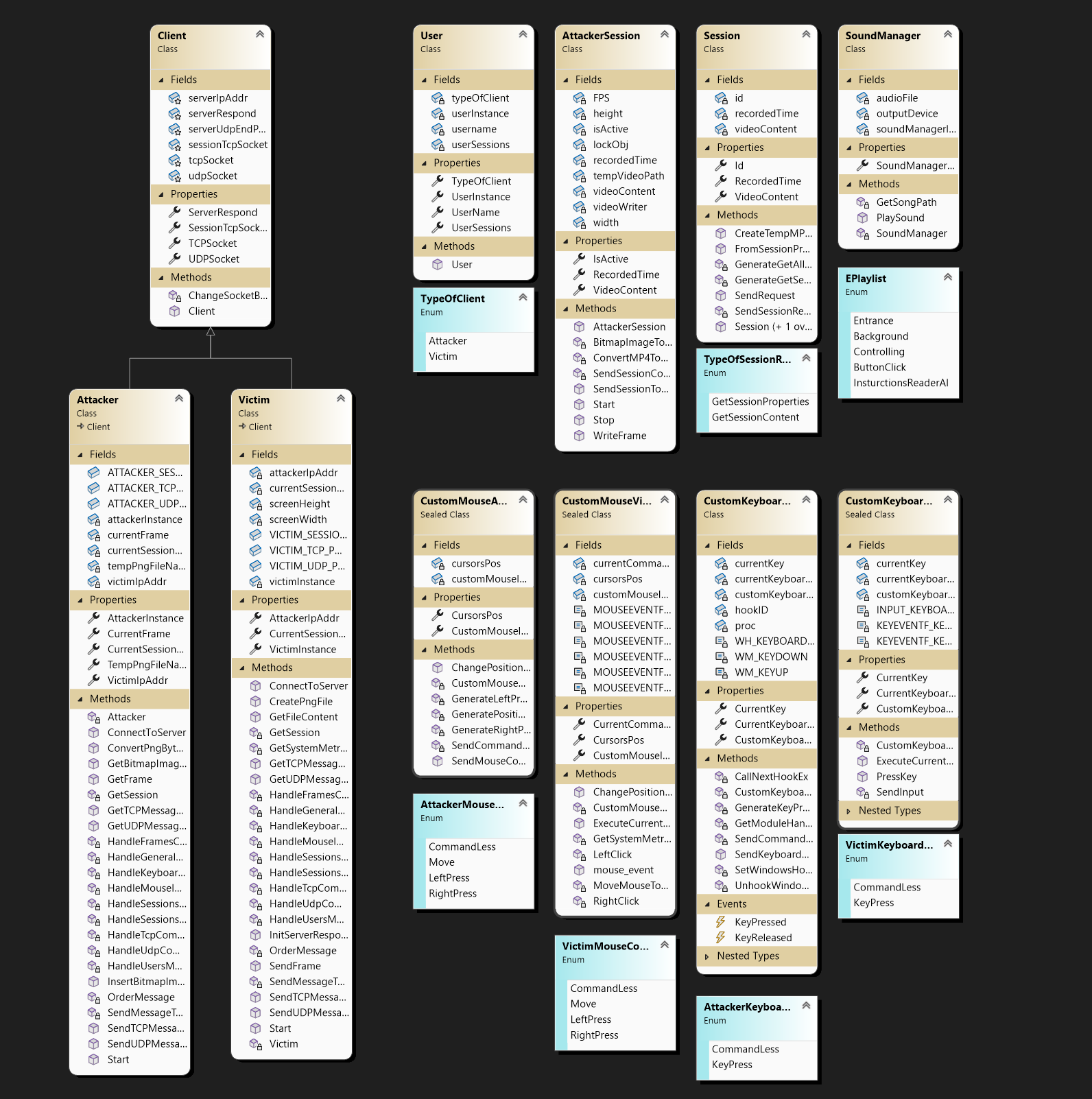
הטכנולוגיות שבהן השתמשתי בעת יצירת הפרויקט הינן: שפת התכנות C#, פלטפורמת הפיתוח WPF וטכנולוגית אחסון הנתונים SQL.

**תרשים UML-**

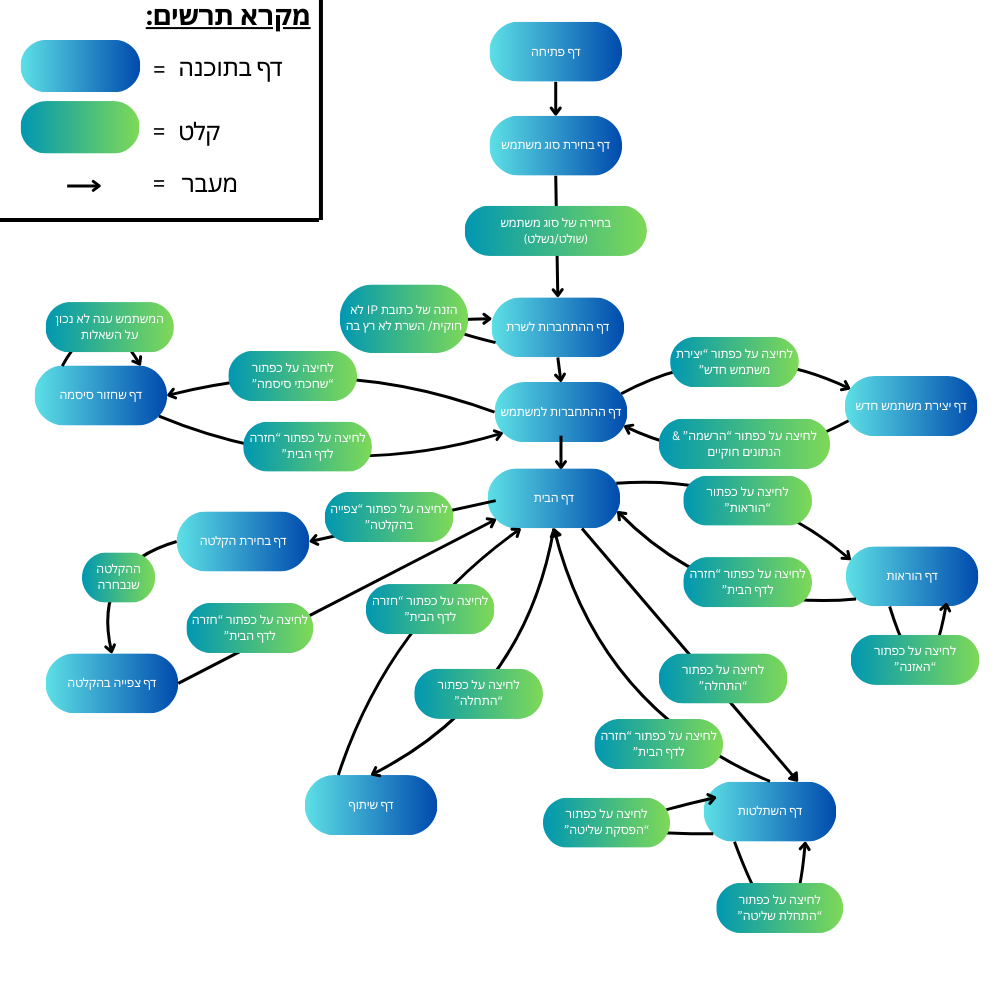
**צד שרת-**

****

**צד לקוח-**

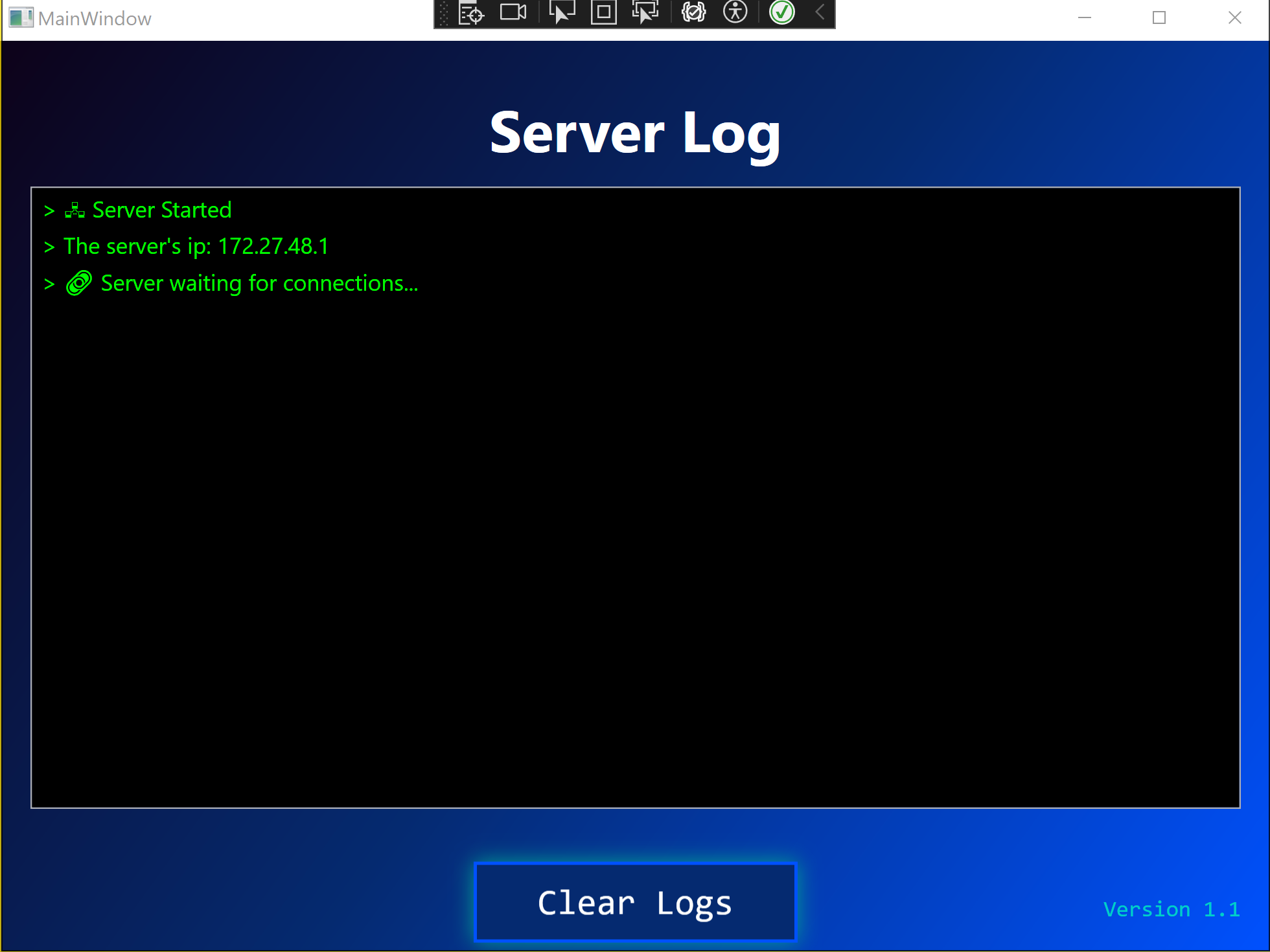
****

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:**

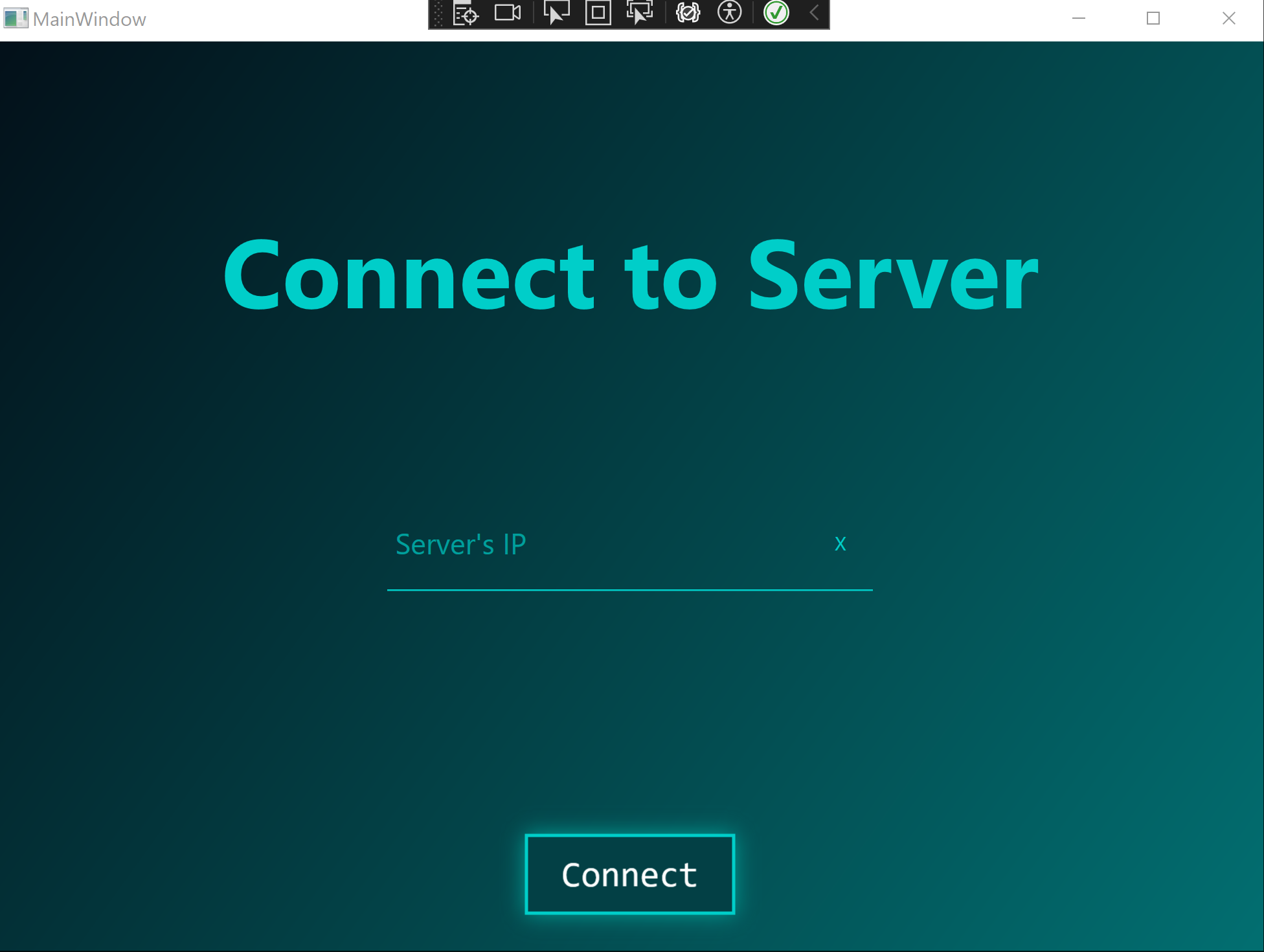
כעת, יש להפעיל את תוכנת הלקוח הנקראת “ExtremLink\_Client\_v2”. תחילה תופיעה אנימצית הפתיחה ומשם התוכנה תעבור לדף בחירת סוג המשתמש. בדף זה, על הלקוח לבחור (באמצעות לחיצה על התמונות המופיעות) האם ברצונו להיות **השולט** בלקוח אחר או זה ש**נשלט** על ידי לקוח אחר.

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 הודעת השגיאה הזאת: