



# Cyber Project

## TK Cinema

שם התלמיד: תומר קסורלה.

ת.ז.: 323977504.

ביה"ס: תיכון חדש - הרצליה.

שם המנחה: עדינה אנדן.

# תוכן עניינים

## תוכן עניינים

3.....	מבוא
7.....	מבנה \ ארכיטקטורה
24.....	בסיס הנתונים
25.....	מדריך למשתמש
30.....	מדריך למפתח
31.....	רפלקציה
32.....	ביבליוגרפיה
32.....	נספחים

# מבוא

## הרקע לפרויקט

הפרויקט שאותו בחרתי לעשות הוא מערכת המספקת שירותי צפייה ישירה. במערכת ישנם עוד הרבה פיצ'רים נוחים ושימושיים למשתמשים. המערכת מאפשרת הרשמה וכניסה, צפייה בסרטונים הנמצאים בה בסטרימינג ועוד הרבה אפשרויות נוספות שיפורטו בהמשך. הבעיה העיקרית שעליה עונה הפרויקט שיצרתי היא שכיום כלל האנשים בעולם משלמים כסף לחברות המעניקות שירותי צפייה ישירה כמו יוטיוב או הוט כל חודש. בנוסף לכך משלמים על שירותי הצפייה הישירה צריך לרכוש ממיר או טלוויזיה חכמה על מנת לקבל את השירות שנתון לתלות בו נמצאת הטלוויזיה. המערכת שיצרתי עובדת בנוחות מרבית ונותנת את שירותי הצפייה הישירה בחינם ויכולה לפעול על כל מחשב המחובר לאינטרנט ללא תלות במקום בו אתה נמצא. בנוסף המערכת משדרת את התכנים המבוקשים בסטרימינג וכך לא מבזבזת מקום בזיכרון המחשב.

## תהליך המחקר הראשוני

המחקר שלי אודות הפרויקט החל לפני שבחרתי בו על מנת להחליט באיזה נושא אבחר ואיזה דברים אממש. ראשית, מכיוון שהתעניינתי בנושא זה סקרתי את המצב הקיים בשוק והסתכלתי על כל מיני תוכנות המובילות בתחום הצפייה הישירה ולקחתי מהן השראה ורעיונות לפיתוח. בנוסף לכך, חשבתי רבות על חידושים מעניינים ושימושיים שאוכל לעשות במערכת שלי שאין בשוק והגעתי לכך שהמערכת שלי היא חנימית ואינה מוצפת בפרסומות שמעיקות על המשתמש. חידוש נוסף שחשבתי עליו הוא לאפשר למשתמש להוריד את הסרטון ולצפות בו דרך המערכת שלי ללא חיבור לאינטרנט בכל זמן שרק ירצה ולדרג את הסרטון על פי רצונו. בנוסף לכך, בדקתי על ספריות רבות בפייתון שמציגות ומשמיעות סרטונים, אך לא מצאתי ספריה שיודעת לפתוח קובץ וידאו עם שמע (כמו MP4) ולחלק אותו לחלקים כדי שאוכל לבצע סטרימינג, כלומר, להעביר את החלקים ברשת ולנגן אותם בכל לקוח. המשכתי בחיפוש ומצאתי שתי ספריות, אחת ששמה cv2 והיא מחלקת קובץ וידאו לפריימים והשנייה pyaudio והיא מחלקת קובץ אודיו לצ'אנקים. לאחר שחקרתי על שתי ספריות אלה, הבנתי שכדאי לי לעבוד עם קבצים בפורמט לא דחוס כדי שהפריימים והצ'אנקים יהיו פחות גדולים וכך יהיה יותר

יעיל להעביר אותם ברשת. הרעיון שהיה לי הוא לחתוך קובץ של AVI לחלקים של 10 שניות באמצעות תוכנה חיצונית שנקראת FFMPEG ולהעביר את החלק של הסרטון ללקוח ואז מכל חלק של סרטון להוציא את השמע לקובץ WAV באמצעות אותה תוכנה חיצונית ולנגן גם את הוידאו וגם את האודיו באמצעות הספריות, אך מכיוון שזה פותח כל קובץ מחדש ומנגן אותו היה תקיעות לאחר כל חלק שהסתיים. אז הפתרון הסופי שלי היה להעביר כל פריים וכל צ'אנק שיוצרות הספריות בהן בחרתי להשתמש.

## **הסיבות לבחירת הנושא**

בחרתי בפרויקט זה מכיוון שנושא זה נורא עניין אותי ורציתי לחקור אותו ולהתעמק בו. היה לי חשוב מאוד למצוא פרויקט שיעניין אותי בצורה בלתי רגילה ויגרום לי להנאה כאשר אני עובד עליו ולכן לאחר חקירת הנושא בחרתי בנושא זה. הרעיון הראשוני שלי היה למצוא פתרון לצפייה של סרטונים בצורה נוחה ללא תלות במקום בו אתה נמצא או לצורך טלוויזיה וממיר מסוים כמו HOT, YES. לאחר מכן החלטתי לפתח את הרעיון ולאפשר צפייה בסטרימינג, ובעצם לבנות מערכת שהשרת בה מכיל קבצי וידאו שהועלו אליו ומאפשר למשתמשים לצפות בסרטונים בסטרימינג (ללא להוריד את קבצי הוידאו למחשב ולתפוס זיכרון). לאחר מכן החלטתי גם להוסיף אפשרות של צפייה ללא חיבור לאינטרנט בעזרת הורדת הסרטון למחשב בלחיצת כפתור ולאפשר צפייה בסרטון בעזרת המערכת שלי. המערכת מעבירה את הזמן בכיף, בצורה מעניינת ומציגה מגוון רחב של סרטונים ואפשרויות. המוטיבציה לפרויקט שלי הייתה גבוהה מאוד לאורך כל הדרך מכיוון שנושאי המחקר הרבים שלקחתי על עצמי לחקור, ההתמודדות איתם והצלחתם גרמו לי לסיפוק ולהרבה הנאה מהתכנות. בנוסף לכך, היו נושאים קשים שלקח לי הרבה זמן ומחשבה לפתור וכאשר הצלחתי אותם זה נתן לי מוטיבציה רבה וגרם לי להמשיך קדימה וללמוד עוד הרבה דברים חדשים.

## **נושאי מחקר והתמודדות**

במהלך ביצוע הפרויקט חקרתי נושאים רבים והתמודדתי עם קשיים רבים:

- **סנכרון הוידאו והאודיו בכל סרטון** - ראשית, הבנתי שהאודיו קבוע ולפיו צריך לסנכרן את הוידאו שמתנגן מהר יותר מהאודיו. הפתרון היה להתחיל סטופר כאשר מתחילים את האודיו ולגשת לכל קובץ וידאו ולקחת ממנו את הנתון FPS – frame per second ודרכו לחשב את הזמן שלוקח כל FRAME וכך לחשב את הזמן בו הוידאו נמצא. כאשר יש לי

את הזמנים גם בוידאו וגם באודיו החלטתי להוסיף בפונקציה שבה אני מציג את הוידאו בדיקה אם הזמן בוידאו יותר גדול מהזמן באודיו אז לעשות שינה קצרה.

- **העברת CHUNKS של אודיו באמצעות פרוטוקול UDP בסטרימינג מהשרת ללקוחות** – הספריה pyaudio יוצרת צ'אנקים מסוג ביטים שניתן להעביר בקלות בsocket.
- **העברת FRAMES של וידאו בסטרימינג מהשרת ללקוחות** – הספריה שבה בחרתי להשתמש cv2 יוצרת אובייקט של פריים שאותו ניתן להציג באמצעותה, בשונה מהספריה pyaudio שיוצרת צ'אנק מביטים שאותם ניתן להעביר בקלות בsocket. השתמשתי בספריה pickle על מנת להעביר את הפריים בudp-socket, אך זה לא הצליח מכיוון שפרוטוקול זה אינו אמין ואינו מבטיח הגעה של כל המידע באותו הסדר שבו נשלח. הבנתי שעליי להשתמש בtcp ומצאתי ממשק ששמו zmq המעביר מידע בצורה יעילה בsocket ובו החלטתי להשתמש לאחר שראיתי שזה מעביר בצורה מהירה יותר מאשר להשתמש בpickle.
- **נושא ה GUI בWFA** – תקשורת עם התהליך בפיתוח, הרצת התהליך בפיתוח ויצירת GUI דינאמי על פי הנתונים שמקבל.
- **ספריית sqlite3** – יצירת database דינאמי המאפשר הוספת משתמשים, סרטונים, זמני עצירת הסרטון והסרטים שהוריד לכל משתמש, ודירוג הסרטונים.
- **ספריית multithreading** – מקביליות בין דברים הצריכים לרוץ במקביל.
- **הצפנות RSA AES** – הצפנת התעבורה בין השרת והלקוחות.
- **הרצה קדימה** – חישובתי את כמות הצ'אנקים שמושמעים ב10 שניות ועל ידי כך אני מדלג על כמות זו, ובמקביל מדלג גם על כמות הפריימים בעשר שניות ומוסיף עשר שניות לזמן הנוכחי של הסרטון.
- **נגינה מנקודת הצפייה האחרונה** – מכיוון שלוקח זמן עד שמעבירים את כל המידע עד לנקודה זו חישובתי את הזמן שצריך לקחת עד להגעת זמן זה של הסרטון ללקוח ורק לאחר זמן זה יתחיל הסרטון.
- **נושא העברת התעבורה בTCP SOCKET בדרך היעילה ביותר** – בתהליך המחקר שעשיתי גיליתי שתי דברים חשובים שגרמו להעברת התעבורה שלי להיות מהירה ויעילה מאוד באופן משמעותי. ראשית, כדי להעביר מידע ארוך (כמו תמונה או סרטון) השתמשתי בפונקציה sendall שמאפשרת שליחה של כמויות גדולות של מידע באופן היעיל ביותר. בנוסף, על מנת לקבל את המידע באופן היעיל ביותר, הצד השולח מעביר לצד המקבל את אורך המידע שהוא רוצה להעביר. לאחר מכן, הצד המקבל עושה recv לאורך המידע שאמור לקבל פחות אורך המידע שכבר קיבל, ובצורה זו מקבל את כמות המידע המקסימלית בכל קריאה לפונקציה.

- **אוטומציה של הורדת סרטון** – על מנת לאפשר דרך נוחה ושימושית להוריד סרטונים לשרת יצרתי סקריפט שמקבל רשימה של urls של סרטונים ומוריד אותם באמצעות ספריית pytube. לצערי ספרייה זו לא עובדת יותר מכיוון שיוטיוב לא מאפשרים זאת מסיבות של זכויות יוצרים.

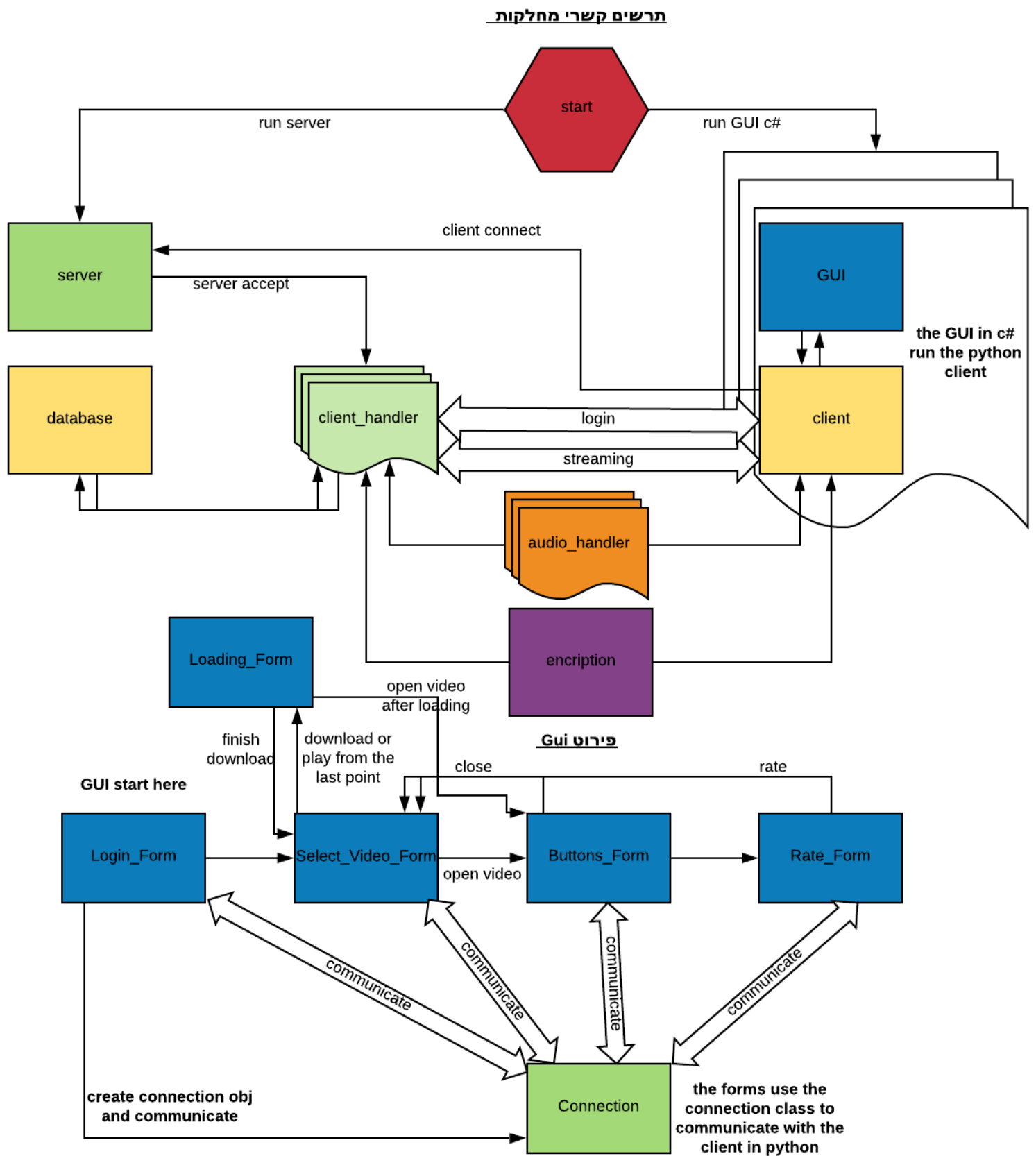
## **קהל היעד**

קהל היעד של המערכת שלי הוא נרחב וכולל כל בנאדם שברשותו מחשב המחובר לאינטרנט. כאמור, המערכת מתאימה לכל שכבות הגיל באוכלוסייה ובמיוחד לכל בנאדם שאינו מעוניין לשלם כל חודש על מנת לקבל שירותי צפייה ישירה או לרכוש ממיר או טלוויזיה חכמה. בנוסף, כל בנאדם המעוניין לצפות בסרטונים בצורה נוחה וללא תלות במקום בו הוא נמצא. המערכת מציעה אפשרות לצפייה ללא חיבור לאינטרנט ומתאימה את עצמה לאנשים שרוצים לראות את הסרט במועד מאוחר יותר ללא חיבור (למשל בטיסה).

# מבנה \ ארכיטקטורה

## הצגת המערכת

צד ה-server של הפרויקט כתוב בשפת התכנות python. כמו כן, גם ה-"מנוע" של צד ה-client כתוב בשפת התכנות python. ממשק המשתמש, ה-GUI, ייכתב בשפת: C#, ב-IDE: Visual Studio, בפרויקט מסוג: Windows Forms App. המערכת תומכת במספר קליינטים שיכולים להתחבר במקביל, מאפשרת גם רישום לקוחות חדשים וכניסה ללקוחות רשומים שאת פרטיהם תשמור ב-DATABASE. לאחר ההרשמה המערכת מציגה את כל התמונות של הסרטונים שנשלחים אליה לאחר כל התחברות מהשרת ומאפשרת לבצע חיפוש לפי שם הסרטון וכמה פעולות על כל סרטון. ניתן לצפות בו בסטרימינג, ניתן לצפות בו מנקודת הצפייה האחרונה שנשמרת ב-DATABASE עבור כל משתמש שמתחבר למערכת במידה וצפה בסרטון ויצא לפני שנגמר, ניתן להוריד את הסרטון ולצפות בו ללא חיבור בשלב מאוחר יותר וניתן לראות את דירוג הסרטון שגם הוא שמר ב-DATABASE עבור כל סרטון. לאחר שהמשתמש בוחר באחת האפשרויות נפתח הסרטון שבחר וביחד איתו חלון של שליטה שמאפשר עצירה, השתקה, קפיצה של 10 שניות קדימה, ויציאה. אם המשתמש הגיע לסוף הסרטון יפתח לו חלון שמאפשר דירוג של הסרטון. לאחר הדירוג או לאחר שהמשתמש סגר את הסרטון באמצע יפתח שוב חלון בחירת הסרטונים.





## פירוט מחלקות הפרויקט

### מחלקות שרת:

**Server** - מחלקה אשר יוצרת את הסרבר ואת תכונותיו הבסיסיות. המחלקה יוצרת את האובייקט database שיוצר קובץ אם אין כזה. בנוסף המחלקה מקימה socket המבוסס tcp ומחכה שלקוח התחבר אליו. כאשר לקוח מתחבר נוצרים מפתחות הצפנה שמועברים לקליינט על מנת שהתעבורה ביניהם תהיה מוצפנת. המערכת מאפשרת התחברות דינאמית של לקוחות ומקימה עבור כל לקוח אובייקט מסוג CLIENT HANDLER המטפל בו.

**ClientHandler** – מחלקה זו יורשת את multithreading.Thread, לכל קליינט שיתחבר למערכת יוקם אובייקט מסוג מחלקה זו אשר יגדיר את תכונותיו ויאפשר תקשורת במקביל של הסרבר עם כל אחד מלקוחות המחוברים. בנוסף הקליינט הנדלר ינהל את כל נושאי ההתחברות של משתמש, הסטרימינג של הוידאו והאודיו, הורדת סרטון וכתביה וקריאה מהDATABASE.

**Database** – במחלקה זו נוצרים שני tables שבאחד נמצאים שמות המשתמשים סיסמאות, הזמנים שבהם הפסיק את הצפייה והסרטים שהוריד ובשני שם הסרטון, דירוג ומספר מדרגים. במחלקה זו יש ניתן להוסיף משתמש, לבדוק אם משתמש נמצא, להוסיף סרט, לבדוק אם הסרט נמצא, לקבל רשימה של שמות הסרטונים, לקבל דירוג של סרט, לקבל את הזמנים שבהם הפסיק את הצפייה בסרטונים, לעדכן את זמני הצפייה, לעדכן דירוג, להוסיף סרט שהוריד, לקבל את הסרטים שהוריד.

**סקריפט עזר להורדת סרטונים לשרת** - הסקריפט מקבל קובץ avi לתיקה מוגדרת ויוצר קובץ wav שמכיל את האודיו בקובץ avi ושומר אותו בשמו של קובץ avi.

### מחלקה משותפת:

**AESEncryption ו RSACrypt** – מחלקות אלה דואגות להצפנת המידע המועבר ברשת בין השרת ללקוחות וכן פיענוח של אותו מידע העובר בעת הגעתו ליעד הרצוי.

**Audio\_handler** – מחלקה זו מאפשרת חלוקה לCHUNKS של אודיו וכתביה שלהם לכרטיס הקול במחשב, בנוסף מאפשרת המחלקה להשתמש באובייקט משותף שצריך בשני הצדדים על מנת שהאודיו יעבוד.

**Path\_class** – מחלקה המכילה את כתובת התיקה של הפרויקט לכל הקבצים שצריכים בה שימוש

### מחלקות לקוח:

**Client** – הקובץ בו יוצרים אובייקט מסוג מחלקה זו מורץ על ידי התהליך ב-C#. מחלקה זו מתחברת לשרת ולתהליך ב-C# על ידי סוקט ומקבלת מהשרת נתונים כמו מפתחות הצפנה, הפורטים לתקשורת ותמונת של הסרטונים הנמצאים במערכת. המחלקה מקבלת מה-C# הודעה של שם הקובץ שנבחר על ידי המשתמש או הפעולה שרוצים לעשות עליו, מתנהלת בהתאם למקרה המבוקש ושולחת את ההודעה גם לשרת על מנת שהתנהל בהתאם. הפעולות שיכולות להתבצע הן:

- הרשמה או כניסה – ניהול ההרשמה או הכניסה של לקוח רשום למערכת בשיתוף עם השרת הבודק את הנתונים שב-DATABASE.
- שידור בסטרימינג - שידור בסטרימינג של שם הקובץ המבוקש מהשרת ללקוח.
- צפייה מנקודת הצפייה האחרונה – שידור בסטרימינג של שם הקובץ המבוקש מהשרת ללקוח, אך הלקוח מחכה עד שהנקודה המסוימת תגיע ורק אז מציג את הסרטון המבוקש.
- הורדת הסרטון – העברה של הסרטון מהשרת ללקוח
- נגינת הסרטון ללא חיבור – נגינה של הקובץ שנמצא על המחשב ללא התערבות של השרת למעט קבלת הדירוג ונקודת הצפייה האחרונה.

### מחלקות ה-GUI ב-C#:

**Login\_Form** – מחלקה המורצת בהתחלה ופותחת חלון של הרשמה וכניסה למערכת. המחלקה יוצרת אובייקט מסוג מחלקת connection על מנת לפתוח תקשורת עם תהליך הלקוח בפיתוח באמצעות socket ושולחת את הפרטים שהמשתמש הכניס על מנת לבדוק שהם אכן מתאימים.

**Connection** – מחלקה הדואגת לתקשורת בין התהליך ב-C# לבין התהליך בפיתוח בעזרת socket ובנוסף דואגת להריץ את תהליך הפיתוח.

**Select\_Video\_Form** – מחלקה הדואגת לניהול חלון שמאפשר בחירה של הסרטונים שקיימים במערכת על ידי הצגת התמונות שמתארות כל סרטון ומאפשרת לבחור מספר

פעולות על כל סרטון (צפייה בסטרימינג, הורדת הסרטון, צפייה ללא חיבור וצפייה מנקודת הצפייה האחרונה). בנוסף לכך, מחלקה זו גם מאפשרת לחפש סרטונים על פי שמם. הנתונים נשלחים בעזרת socket שמתפעלת מחלקת connection.

**Buttons\_Form** – מחלקה הדואגת לניהול חלון שמאפשר שליטה בסרטון הניצפה על ידי הלקוח (הרצה קדימה, עצירה והשתקה). בנוסף לכך, המחלקה גם דואגת לפתוח את חלון הדירוג אם הלקוח סיים לצפות בסרטון.

**Rate\_Form** - מחלקה הדואגת לניהול חלון דירוג על ידי כוכבים בין 1 ל 5. הדירוג נשלח ללקוח ששולח לשרת שמעדכן ב-DATABASE. לאחר הדירוג נפתח שוב החלון שמאפשר בחירה של הסרטונים.

**Loading\_Form** – מחלקה הדואגת לניהול חלון טעינה שנפתח כאשר מורידים את הסרטון למחשב או כאשר צופים מנקודת הצפייה האחרונה והסרטון נטען.

**Project\_path** – מחלקה המכילה את כתובת התיקיה של הפרויקט לכל הקבצים שצריכים בה שימוש

## תפעול הפרויקט

הפרויקט מבוסס על קבצי וידאו בפורמט לא דחוס (AVI), אותם הוא קורא פריים פריים ומשדר בהתאם. כדי לשדר את פס הקול, ברגע שהקליינט מעלה סרטון לשרת בפורמט AVI יופרד ממנו השמע לקובץ נפרד של WAV באמצעות סקריפט שכתבתי שמשתמש בתוכנה חיצונית הנקראת ffmpeg ושומר את הקובץ באותו השם של קובץ ה-AVI. שידור זה יעשה במקביל באמצעות THREADS. ה chunks של האודיו ישלחו באמצעות פרוטוקול UDP, שיאפשר החמצה של מידע. ה frames של הוידאו (שהם אובייקטים) ישלחו בממשק הנקרא ZMQ המבוסס על תקשורת TCP ומאפשר לשלוח מידע בקידודים שונים (כמו למשל אובייקטים) בצורה יעילה. מתבצע סנכרון על ידי חישוב של הזמן מנתוני סרטון הוידאו בהשוואה לאודיו שהוא קבוע ונמדד באמצעות סטופר. נתוני המשתמשים ונתוני הסרטונים יישמרו ב-DATABASE.

## הסבר מפורט על העברת הוידאו

בחרתי להשתמש בממשק ZMQ לאחר ניסיונות רבים של בדיקת יעילות של הסטרימינג. ראשית כשניסיתי להעביר ב-UDP את ה-FRAMES נתקלתי בבעיה שהאובייקט FRAME

שמקבלת הספרייה כדי להציג פריים מסוים אינו מגיע בשלמותו (גם בגלל שUDP לא מבטיח על סדר הגעת הפקטות וגם בגלל שלא דואג שכולן יגיעו). הבנתי שהספרייה שאני משתמש בה אינה יכולה להציג תמונה בלי כמה פיקסלים שחסרים אלא רק אובייקט פריים שמגיע בשלמותו. ולכן הבנתי שאני צריך להעביר את הפריים בTCP וחפשתי את הדרך היעילה ביותר לעשות זאת (מכיוון כשחילקתי את המידע לפקטות והעברתי אותו זה לא עבר במהירות מספקת). מצאתי את הממשק ZMQ ובחנתי אותו בין שני מחשבים וראיתי שהוא עובד בצורה יעילה יותר מאשר להעביר את המידע עם PICKLE.

### **תיאור מפורט של התרחישים עליהם עונה המערכת:**

ראשית ירוץ השרת שיוצר socket, מפתחות הצפנה ויוצר קובץ DATABASE במידה ולא קיים ולאחר מכן מחכה להתחברות של לקוחות. כאשר מריצים את הGUI נפתח חלון הרשמה/ כניסה שיוצר socket, לאחר מכן מריצים את תהליך של הקליינט בפיתון. תהליך הקליינט בפיתון מתחבר לשרת ומחליפים מפתחות הצפנה בצורה הבאה:

ראשית, השרת שולח את מפתח rsan הציבורי שיוצר. לאחר מכן הקליינט מקבל את המפתח הציבורי ויוצר מפתח aes שאותו שולח לשרת מוצפן בעזרת מפתח rsan הציבורי. השרת מקבל את המפתח המוצפן ומפענח אותו בעזרת מפתח rsan הפרטי שיצר, וכך מתנהלת כל התעבורה ברשת בהצפנה עם מפתח aes שמצפין במהירות ויעילות לעומת rsan. לאחר מכן שולח השרת רשימת פורטים עליהם תתבסס התקשורת מול הקליינט הנדלר ויוצר את האובייקט קליינט הנדלר שמטפל בכל קליינט שמתחבר לכוד. לאחר מכן הקליינט הנדלר מתחיל לשלוח ללקוח את התמונות המתארות את הסרטונים שנמצאים במערכת אם אין ללקוח אותם ובנוסף בודק אם הם הסרטון נמצא בDATABASE ואם לא מוסיף אותו, הלקוח מקבל אותם ושומר אותם בתיקיה שיוכל להשתמש בהם בGUI בחירת הסרטונים. לאחר מכן הלקוח יוצר socket ומתחבר לתהליך של הGUI בC#. לאחר מכן הGUI מאפשר הכנסת פרטים של המשתמש וישלח ללקוח את פרטיו. הלקוח יוצר אובייקט קליינט ומקבל את הפרטים שהמשתמש הכניס מהsocket בC#. הלקוח בודק אם המשתמש רוצה להירשם או להתחבר- אם הוא רוצה להתחבר הלקוח שולח את פרטיו לשרת ובודק שהם אכן מתאימים, אם הוא רוצה להירשם הלקוח בודק שהסיסמא מספיק חזקה, כלומר בודק שמכילה אותיות קטנות, גדולות ומספרים אם הוא מכיל שולח את זה לשרת ורושם את המשתמש למערכת אם השם משתמש לא תפוס אחרת שולח לC# הודעות המכילות מה לא בסדר בסיסמא. השרת יכניס לDATABASE במקרה של רישום או יבדוק את הפרטים של המשתמש במקרה של כניסה. לאחר מכן הקליינט הנדלר שולח ללקוח מחרוזת שאוסף את

נתוניה מהDATABASE הכוללת - כל סרט ונקודת הצפייה האחרונה שלו, הסרטים שהלקוח הוריד והדירוג של כל סרט. הלקוח מקבל את המחרוזת הזו ושולח לC# ושם הוא לוקח את המידע ושומר אותו במבני נתונים (שני מילונים ומערך של סטרינגים) ולאחר מכן יציג את הנתונים כאשר הצטרך. לאחר מכן נפתח חלון בחירת הסרטונים הכולל מספר תמונות של הסרטונים הקיימים במערכת, בחלון זה ניתן לחפש סרטונים על פי שמם, כלומר, כל תו שהכתב בתיבת חיפוש ישלח לשרת ושם ייבדק איזה שמות מהסרטים מתאימים. הסרטונים המתאימים ישלחו לGUI ותמונותיהם בלבד יוצגו על המסך. בנוסף לכך, חלון זה מאפשר לערוך על כל סרטון מספר פעולות על ידי לחיצה על לחצן ימין של העכבר:

לנגן בסטרימינג, לנגן מנקודת הצפייה האחרונה, להוריד אותו, ולצפות בו ללא חיבור במידה והורדת. לאחר שבחרים באחת מהאפשרויות הבחירה נשלחת ללקוח שהוא שולח אותה לקליינט הנדלר ומתנהל בהתאם.

הרחבה על כל אחת מהאפשרויות:

- נגינה בסטרימינג – כאשר המשתמש בוחר באפשרות זו נשלחת הודעה זו ללקוח ומיד הוא מעביר אותה לשרת. הלקוח פותח שני threads:
  - thread הראשון לוידאו שבו נפתחים עוד שני threads. thread אחד שמקבל את כל הפריימים באמצעות ממשק הנקרא ZMQ המבוסס על תקשורת TCP ומאפשר לשלוח מידע בקידודים שונים (כמו למשל אובייקטים) בצורה יעילה והשני שמציג אותם באמצעות הספריה cv2.
  - thread זה גם דואג לסנכרון הוידאו והאודיו. ראשית, מכיוון שהאודיו קבוע אז לפיו צריך לסנכרן את הוידאו שמתנגן מהר יותר מהאודיו. הפתרון הוא להתחיל סטופר כאשר מתחילים את האודיו ולגשת לכל קובץ וידאו ולקחת ממנו את הנתון FPS – frame per second – ודרכו לחשב את הזמן שלוקח כל FRAME וכך לחשב את הזמן בו הוידאו נמצא. כאשר יש את הזמנים גם בוידאו וגם באודיו נוצרת בדיקה בthread שבו מציגים את הוידאו האם הזמן בוידאו יותר גדול מהזמן באודיו, אם כן אז לעשות שינה קצרה. בנוסף, thread זה גם פותח thread שאחראי על לקבל פקודות מהGUI כמו עצירה, סגירה, השתקה, וקפיצה של 10 שניות קדימה. thread הצגת הסרטון אחראי גם על עצירת הסרטון כאשר מתקבלת הודעת עצירה מthread שמקבל פקודות מהGUI. thread הצגת הסרטון אחראי גם על לדלג על מספר פריימים שחושבו כאשר המשתמש מבקש לקפוץ קדימה 10 שניות והוא נסגר כאשר מתקבלת פקודת הסגירה מthread פקודות מהGUI.

- thread השני לאודיו שבו נפתחים גם עוד שני threads. thread אחד שמקבל את הצ'אנקים באמצעות udp socket והשני שמנגן אותם באמצעות כתיבה לכרטיס קול עם ספריית pyaudio. thread זה אחראי גם על להשתיק את הסרטון, ולעצור אותו – פקודות אלה מתקבלות מהthread שמקבל פקודות מהGUI. בנוסף לכך, thread זה אחראי על לשלוח לGUI הודעה כאשר מסתיים הסרטון וכך לפתוח את חלון הדירוג. כאשר המשתמש יצא באמצע הסרטון thread זה אחראי על לשלוח את הזמן לקליינט הנדלר כדי שיעדכן את הזמן בDATABASE. בנוסף לכך, כאשר מקבלים פקודה להקפצה קדימה של 10 שניות thread זה אחראי על לדלג את כמות הצ'אנקים שחושבו ולהוסיף לזמן של האודיו 10 שניות. thread נסגר כאשר מתקבלת פקודת הסגירה מthread פקודות מהGUI.
- במקביל בשרת נפתחים שלושה threads, אחד לוידאו שדואג לקרוא מהקובץ פריים אחר פריים ולהעביר אותו באמצעות ממשק ZMQ, והשני לאודיו שדואג לקרוא מהקובץ צ'אנק אחר צ'אנק ולהעביר אותו באמצעות udp socket. thread זה אחראי גם על שליחת הודעה לthread המקבל את הצ'אנקים שנגמר הסרטון כאשר נגמר הסרטון, כלומר לא נשאר עוד צ'אנקים שלא נשלחו בקובץ. thread השלישי אחראי על קבלת הודעת דירוג או זמן יציאה מהלקוח ומכניס נתונים אלה לDATABASE.
- נגינה מנקודת הצפייה האחרונה - כאשר המשתמש בוחר באפשרות זו נשלחת הודעה זו ללקוח ומיד הוא מעביר אותה לשרת. גם פה מתנהל אותו תהליך של הסטרימינג שצוין למעלה, אך בנוסף, נוצר חישוב של הצ'אנקים והפריימים שצריך לדלג והזמן שלוקח לשרת להעביר את המידע על מנת לנגן מנקודה זו. המערכת מתנהלת על פי הדברים שחושבו כלומר מדלגת על כמות הפריימים והצ'אנקים ורק אחרי שעבר הזמן שחושב מציגה ומנגנת את הסרטון.
- הורדת סרטון - כאשר המשתמש בוחר באפשרות זו נשלחת הודעה זו ללקוח ומיד הוא מעביר אותה לשרת. השרת פותח את קובץ avi של הסרטון המבוקש ושולח אותו באופן מוצפן ללקוח וכך גם את קובץ wav. הלקוח מקבל את המידע שקיבל עבור קובץ avi ומפענח אותו ושומר אותו בקובץ חדש וכך גם את קובץ wav.
- צפייה בסרטון ששמור על המחשב ללא חיבור - כאשר המשתמש בוחר באפשרות זו נשלחת הודעה זו ללקוח ומיד הוא מעביר אותה לשרת. השרת לא עושה כלום מלבד לחכות לדירוג או לנקודת הצפייה האחרונה. הלקוח פותח שני threads:
  - thread ראשון לוידאו שפותח את קובץ avi של הסרטון המבוקש וקורא פריים פריים ומציג אותו. thread זה גם אחראי על הסנכרון שפורט באפשרות הסטרימינג. בנוסף, thread זה גם פותח thread של פקודות

מה GUI ואחראי לעצירה המתקבלת כפקודה thread פקודות מה GUI. thread זה אחראי גם על לדלג על מספר פריימים שחושבו כאשר המשתמש מבקש לקפוץ קדימה 10 שניות. thread נסגר כאשר מתקבלת פקודת הסגירה thread פקודות מה GUI.

- thread השני לאודיו שקורא מקובץ wav של הסרטון המבוקש צ'אנק צ'אנק וכותב אותו לכרטיס הקול. thread זה אחראי על השתקה של הסרטון ועצירתו. thread אחראי גם על שליחת הודעה שנגמר הסרטון ל GUI על מנת שיפתח את חלון הדירוג. thread שולח הודעה שנגמר כאשר לא נשארו צ'אנקים בקובץ שלא ניגן. בנוסף לכך, כאשר מקבלים פקודה להקפצה קדימה של 10 שניות thread זה אחראי על לדלג את כמות הצ'אנקים שחושבו ולהוסיף לזמן של האודיו 10 שניות. thread נסגר כאשר מתקבלת פקודת הסגירה thread פקודות מה GUI.

כפי שציינתי, כאשר לקוח מסיים לצפות בסרט נשלח ל GUI הודעה זו והוא מציג את חלון הדירוג. כאשר המשתמש מדרג דירוג זה נשלח ללקוח והלקוח שולח זאת לקליינט הנדלר כדי שיעדכן את הנתונים ב DATABASE. אם המשתמש סגר את הסרטון לפני שהוא נגמר לא יפתח חלון הדירוג ושילח לקליינט הנדלר הזמן בו סגר את הסרטון על מנת שיוכל להמשיך לצפות ממנו בפעם הבאה. לאחר הדירוג או היציאה מהסרטון באמצע יופיע שוב חלון בחירת הסרטונים וימשיך אותו תהליך עד שהמשתמש יסגור את התוכנה.

## הצגת אלגוריתמים עיקריים:

קובץ **server.py** – החלפת מפתחות ההצפנה עם הלקוח בצד השרת.

```
def encryption_exchange_keys(self, client_socket):
    """
    exchange encryption keys with the client
    :param client_socket:
    :return:
    """
    self.rsa.create_private_key()
    self.rsa.create_public_key()
    client_socket.send(self.rsa.get_public_key())
    msg = client_socket.recv(1024)
    self.aes.set_key(self.rsa.decode(msg))
```

## קובץ `client_handler.py` – שליחת הוידאו והאודיו בסטרימינג והעברת תוכן קבצי

avin ו wav של הסרטון המבוקש מוצפן ללקוח.

```
def handle_audio(self):
    """
    A function that is responsible for streaming audio
    :return:
    """
    py_audio_obj = Audio(self.filename, "")
    receive_list = self.audio_socket.recvfrom(1024)
    msg = py_audio_obj.get_constants()
    self.audio_socket.sendto(msg.encode(), receive_list[1])
    address = receive_list[1]
    print(address)
    message = receive_list[0].decode()
    print(message)
    data = py_audio_obj.readframes()
    while data and not self.close_stream:
        self.audio_socket.sendto(data, address)
        data = py_audio_obj.readframes()
        time.sleep(0.002)
    self.audio_socket.sendto("finish audio".encode(), address)
    py_audio_obj.close()
    print("finish to send the audio")

def handle_video(self):
    """
    A function responsible for streaming the video
    :return:
    """
    cap = cv2.VideoCapture(self.filename + '.avi') # init the camera
    fps = cap.get(cv2.CAP_PROP_FPS) # read from the file the fps => frames per sec
    calc_timestamps = [0.0]
    receive_message = self.video_socket.recv(1024)
    print(receive_message.decode())
    while cap.isOpened() and not self.close_stream:
        try:
            ret, frame = cap.read()
            calc_timestamps.append(calc_timestamps[-1] + 1000 / fps)
            current_time_video = calc_timestamps[-1] / 1000 # Calculate the time taken for each
frame
            time_to_17 = number_to_be_17(str(current_time_video))
            encoded, buffer = cv2.imencode('.jpg', frame)
            jpg_as_text = base64.b64encode(buffer)
            self.video_socket.send(time_to_17.encode())
            self.footage_socket.send(jpg_as_text) # sends tte data as string
            time.sleep(0.002)
        except cv2.error:
            break
    print("finish to send the video")

def send_video_thread(self, name_of_video):
    """
    A function that sends the video information so that the user can download it
    :param name_of_video:
    :return:
    """
    my_file = open(PATH + "songs\\" + name_of_video + ".avi", 'rb')
```



```

data = my_file.read() # read the video file
data_str = b64encode(data).decode('ascii') # makes the data as string
data = self.aes.encrypt_aes(data_str)
self.tcp_socket.send((name_of_video + "$" + str(len(data))).encode())
self.receive_decrypted_message_tcp(1024)
self.tcp_socket.sendall(data)
self.receive_decrypted_message_tcp(1024)
my_file = open(PATH + "songs\\" + name_of_video + ".wav", 'rb')
data = my_file.read() # read the audio file
data_str = b64encode(data).decode('ascii') # makes the data as string
data = self.aes.encrypt_aes(data_str)
self.tcp_socket.send((name_of_video + "$" + str(len(data))).encode())
self.receive_decrypted_message_tcp(1024)
self.tcp_socket.sendall(data)
self.receive_decrypted_message_tcp(1024)
print("finish with transfer the video")

```

**קובץ client.py** – קבלת הוידאו והאודיו בסטרימינג, הצגת הוידאו והשמעת האודיו  
 שהתקבלו בסטרימינג, חישוב של הפריימים והצ'אנקים שצריך לדלג על מנת להגיע לנקודת  
 הצפייה האחרונה וזמן הטעינה, והחלפת מפתחות ההצפנה עם השרת בצד הלקוח.

```

def jump(self, time_audio):
    """
    A function responsible for calculating the amount of chunks and frames
    that need to be skipped to start from the last viewing point
    :param time_audio:
    :return:
    """
    self.jump_counter += int(time_audio / 0.1362) # amount of chunks that need to be skipped
    self.jump_time += self.jump_counter * 0.1362
    self.show = False
    self.count = int(self.jump_counter * 0.7) # amount of frames that need to be skipped
    if self.count % 2 != 0:
        self.count += 1
    self.time_loading = int(
        (time_audio / 3) + 1) # The time it takes for the video to load until it reaches this
point
    print(self.time_loading)

def receive_video_thread(self):
    """
    Function is responsible for receiving the streaming frames
    :return:
    """
    while not self.close:
        video_time = self.video_socket.recv(17).decode()
        if video_time != "":
            video_time = float(video_time)
            frame = self.footage_socket.recv_string()
            img = base64.b64decode(frame)
            np_img = np.frombuffer(img, dtype=np.uint8)
            source = cv2.imdecode(np_img, 1)
            self.video_lock.acquire()
            self.list_of_frames.append(video_time)
            self.list_of_frames.append(source)
            self.video_lock.release()

```

```

def play_video(self):
    """A function that is responsible for displaying the video that is being streamed
    and synchronizing it with the audio"""
    if self.time_loading != 0:
        time.sleep(self.time_loading)
    cv2.namedWindow("frame", 0)
    cv2.resizeWindow("frame", 640, 360)
    self.t_video_commands.start()
    while not self.close:
        if len(self.list_of_frames) > 1:
            self.video_lock.acquire()
            self.current_time_video = self.list_of_frames.pop(0)
            frame = self.list_of_frames.pop(0)
            self.video_lock.release()
            while self.stop:
                cv2.imshow('frame', frame)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            delay = self.current_position_audio_time - self.current_time_video
            if delay > 0.05:
                print(
                    str(self.current_position_audio_time) + "\t" + str(self.current_time_video) +
                    "\t" + str(delay))
            else:
                print("sleep")
                time.sleep(1 / 25)
                self.show = True
            if cv2.waitKey(1) & 0xFF == ord('q'):
                self.close = True
                break
            if delay < 0.4:
                cv2.imshow('frame', frame)
            else:
                self.list_of_frames = self.list_of_frames[self.count:]
    cv2.destroyAllWindows()

def receive_audio_thread(self):
    """
    Function is responsible for receiving the streaming chunks
    :return:
    """
    while not self.close:
        data = self.audio_socket.recvfrom(self.audio_buffer)[0]
        self.audio_lock.acquire()
        self.list_of_chunks.append(data)
        self.audio_lock.release()

def play_song(self, py_audio_obj):
    """
    A function that is responsible for playing the video audio
    :param py_audio_obj:
    :return:
    """
    if self.time_loading != 0:
        time.sleep(self.time_loading)
    start = time.time()
    stop = 0
    while not self.close:
        if self.stop:

```

```

        start_stop = time.time()
        while self.stop:
            pass
        end_stop = time.time()
        stop += (end_stop - start_stop)
    if self.list_of_chunks:
        self.audio_lock.acquire()
        data = self.list_of_chunks.pop(0)
        self.audio_lock.release()
        end = time.time()
        self.current_position_audio_time = (end - start) - stop + self.jump_time
        if data == b'finish audio':
            self.close = True
            self.rate = True
            self.c_sharp_socket.send("finish and open rate".encode())
        else:
            if not self.mute:
                if self.jump_counter > 0:
                    self.jump_counter -= 1
                else:
                    py_audio_obj.write(data)
            else:
                time.sleep(0.135)
    py_audio_obj.close()
    if self.rate:
        rate_from_c_sharp = self.c_sharp_socket.recv(1024).decode()
        self.send_encrypted_message_tcp(rate_from_c_sharp + "&")
        self.rate = False

def encryption_exchange_keys(sock):
    """
    Responsible for replacing encryption keys with the server
    :param sock:
    :return:
    """
    rsa = RSACrypt()
    aes_obj = AESEncryption()
    pickle_public_key = sock.recv(1024)
    rsa.public_key = rsa.unpack(pickle_public_key)
    aes_obj.create_key()
    sock.send(rsa.encrypt(aes_obj.key))
    return aes_obj

```

## **תחומים בהם בפרויקט עוסק:**

### **רשתות**

העברת תעבורה מוצפנת בין השרת ללקוח בעזרת שימוש ב SOCKET. העברת התעבורה מתבצעת באמצעות הפרוטוקולים TCP ו UDP. בנוסף העברת הפריימים בסטרימינג מתבצעת על ידי הממשק ZMQ המבוסס על תקשורת TCP ומאפשר לשלוח מידע בקידודים שונים (כמו למשל אובייקטים) בצורה יעילה.

### **מערכת ההפעלה**

המערכת עובדת על מערכת ההפעלה windows ומשתמשת ב MULTITHREADING כדי ליצור מקביליות וטיפול מקביל בכל אחד מהלקוחות.

### **הצפנות**

הצפנת התעבורה בין השרת ללקוח באמצעות מפתח הצפנה א סימטרי RSA ומפתח הצפנה סימטרי AES. המערכת דואגת להצפין את כל המידע העובר בtcp socket.

### **DATABASE**

מסד נתונים אשר נבנה באמצעות SQL בו יש שני tables. הראשון יכיל תחת כל משתמש את שם המשתמש, הסיסמא ששמורה בhash, הזמנים שבהם הפסיק המשתמש את הצפייה בכל אחד מהסרטונים ושמות הסרטונים שהוריד. והשני יכיל תחת כל סרטון את שמו, את דירוגו ואת כמות האנשים שדירגו אותו.

### **GUI**

בקליינט יהיה GUI באמצעות windows form application שימש להתחברות למערכת, לבחירת הסרטונים שמציאה המערכת, לשליטה בהצגת הסרטון (עצירה, הרצה קדימה ואחורה וכו'), לדירוג כל סרטון ולטעינה של סרטון שיוּרד או שאינו הגיע עדיין לנקודת הצפייה האחרונה שממנה ביקש המשתמש להתחיל.

## **Authentication**

בקליינט שבפייתון נבדקת הסיסמא של כל משתמש המעוניין להירשם למערכת על מנת לוודא שהסיסמא שלו מספיק חזקה כדי לשמור על אבטחת המידע במערכת. המערכת תחזיר הודעות המפרטות על מדוע הסיסמא לא מספיק חזקה ורק כאשר הסיסמא תכיל את כל התנאים ההרשמה תתבצע בהצלחה.

**טבלת תיאור פרוטוקלי התקשרות**

<u>צד שרת</u>	<u>צד לקוח</u>	<u>GUI</u>	<u>הסבר</u>
port*port*port	first/not first	-	השרת שולח ללקוח רשימת פורטים המופרדים ב - *. בתגובה הלקוח שולח first אם זה התחברות ראשונה או not first אם זה לא פעם ראשונה
file_name+\$+data_length  / done	I have it / send the photo / finish with the photoes	-	השרת שולח את שם הקובץ ואת אורך תוכן הקובץ או done אם סיים לשלוח. בתגובה הלקוח שולח I have it אם יש לו את התמונה אחרת send the photo. כאשר הלקוח מסיים להוריד את התמונות הוא שולח finish with the photos .
Ok / The user name already taken / The user name or the password is incorrect	User_name<%>password / User_name<%>password / message	User_name<&>password / User_name<%>password	ה GUI שולח ללקוח את פרטי ההרשמה\ הכניסה. User_name<&> password – להרשמה, User_name<%> password לכניסה. אם משתמש רוצה להתחבר אז הלקוח שולח לשרת User_name<%> password ומעביר את ההודעה שמחזיר ל GUI, אחרת הלקוח בודק אם הסיסמא מתאימה, אם כן שולח לשרת User_name<&> password ומעביר את ההודעה שהחזיר ל GUI, אחרת ישלח הודעה שמסבירה מדוע הסיסמא לא התקבלה ל GUI.
movie_time_str "\$" movies_rates "\$"download_videos	movie_time_str "\$" movies_rates "\$"download_videos	-	השרת שולח ללקוח מאפיינים עבור המשתמש שהתחבר כמו איזה סרטים הוריד, זמן הצפייה האחרון בכל סרט

ודירוג הסרטים כל נתון כזה שולח מופרד עם \$. הלקוח מעביר זאת לGUI			"\$download_videos
הgui שולח את שם הסרט שבחר המשתמש ואת הפעולה שהוא רוצה לבצע עימו. הלקוח מעביר את ההודעה לשרת על מנת שיפעל בהתאם. שם הקובץ לנגינה בסטרימינג, שם וזמן לנגינה מזמן זה, הורדה להוריד סרטון זה, נגינה ללא חיבור כדי לנגן ללא חיבור.	Name / name # time / Download@ name / play_without_connection! name	Name / name # time / Download@ name / play_without_connection! name	-
הסטרימינג נשלח ללא פרוטוקול למעט כאשר נגמר הסרטון נשלח מהthread של האודיו בשרת ללקוח finish audio. בתגובה הלקוח שולח לGUI finish and open rate כדי שיפתח את חלון הדירוג. בתגובה הGUI שולח ללקוח close the wait for commands thread in client כדי לסגור את threadsun שמחכה לפקודה מהמשתמש ושולח את הדירוג שהמשתמש הכניס. הלקוח מעביר לשרת את דירוג המשתמש.	close the wait for commands thread in client / rate	finish and open rate / rate	finish audio
הGUI שולח ללקוח mute כדי להשתיק, stop כדי לעצור, close כדי לסגור, plus כדי לקפוץ 10 שניות קדימה. במקרה שהGUI שלח close אז הלקוח שולח לשרת את הזמן בוא בגר את הסרטון.	Plus / mute / stop / close		

# בסיס הנתונים

## Database

בסיס הנתונים של המערכת הוא database שבו יש שני tables אחד של משתמשים והשני של סרטים.

tablen של המשתמשים מכיל כמפתח ראשי את שם המשתמש כמחרוזת המתקבל מההרשמה, סיסמא בhash כמחרוזת המתקבלת מההרשמה, נקודת הצפייה האחרונה בסרטונים כמחרוזת שמתקבלת כאשר לקוח מסוים הפסיק לצפות בסרט לפני שנגמר והסרטונים שהוריד כמחרוזת שמתקבלים כאשר המשתמש בוחר להוריד סרטון. tablen של הסרטים מכיל כמפתח ראשי את שם הסרט כמחרוזת המתקבל כאשר השרת שולח את התמונות שמתארות את הסרטונים הקיימים ומגלה שהסרטון אינו נמצא ב-databases, את דירוג הסרט כמחרוזת שמתקבל כאשר הלקוח מדרג את הסרטון לאחר שצפה בכולו ואת מספר המדרגים כשלם שגדל לאחר כל דירוג ב1.

### Users table

User name	Hash password	Last view points	Download video

### Movies info table

Movie name	rate	counter



# מדריך למשתמש

## סביבת העבודה

סביבת העבודה שבה כתבתי את רוב הפרויקט שלי בpython היא pycharm, סביבת עבודה זו מאפשרת כתיבה נוחה, סימון שגיאות, השלמות אוטומטיות של פקודות וכן מעיר על צורת קידוד שאינה נכונה. בנוסף השתמשתי בvisual studios כדי לכתוב בC#.

הספריות המרכזיות בפרויקט שלי הן CV2, PYAUDIO שהן ספריות להצגה של סרטונים והשמעת אודיו.

מערכת ההפעלה שתומכת בפרויקט שלי היא של WINDOWS של חברת MICROSOFT בגרסאות 7 ו 10

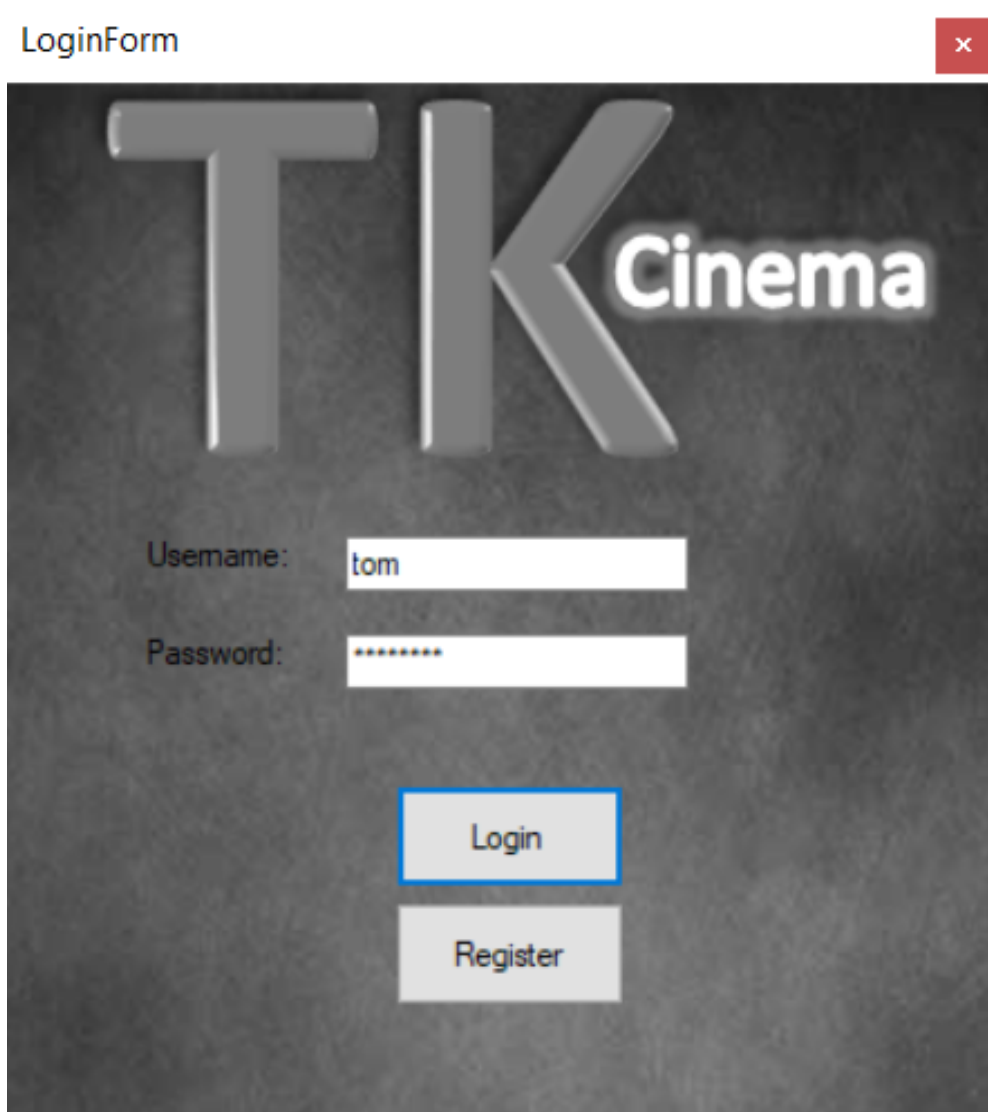
## הוראות התקנה

- סביבת העבודה הנדרשת היא מערכת הפעלה windows שבה מותכן anaconda3
  - visual studio
- הספריות שצריך להתקין בפיתון הן:
  - cv2
    - pip install opencv-python
  - pyaudio
    - pip install pipwin
    - Pipwin install pyaudio
- נדרשת תיקיה ראשית שבה יהיו כל קבצי הפרויקט ובתוכה תיקייה בשם files שבה יהיו תמונות הרקע של הGUI.
- בשרת צריך שתהיה תיקיה ששמה songs ובה יהיו כל הסרטונים במערכת, כלומר קובץ avi, wav ותמונה של הסרטון תחת קובץ bmp.
- על מנת שהמערכת תעבוד על כל מחשב צריך לשנות כמה דברים בקוד:
  - לשנות בקובץ client.py את הIP לקו של השרת.
  - לשנות בקובץ project\_path.py את המשתנה PROJECT\_DIRECTORY לכתובת התיקיה שבה נמצא הפרויקט.
  - לשנות בקובץ project\_path.cs את כתובת הinterpreter של python,
  - את שם כתובת הקובץ client.py ואת הכתובת התיקיה files שנמצאת בתיקית הפרויקט.

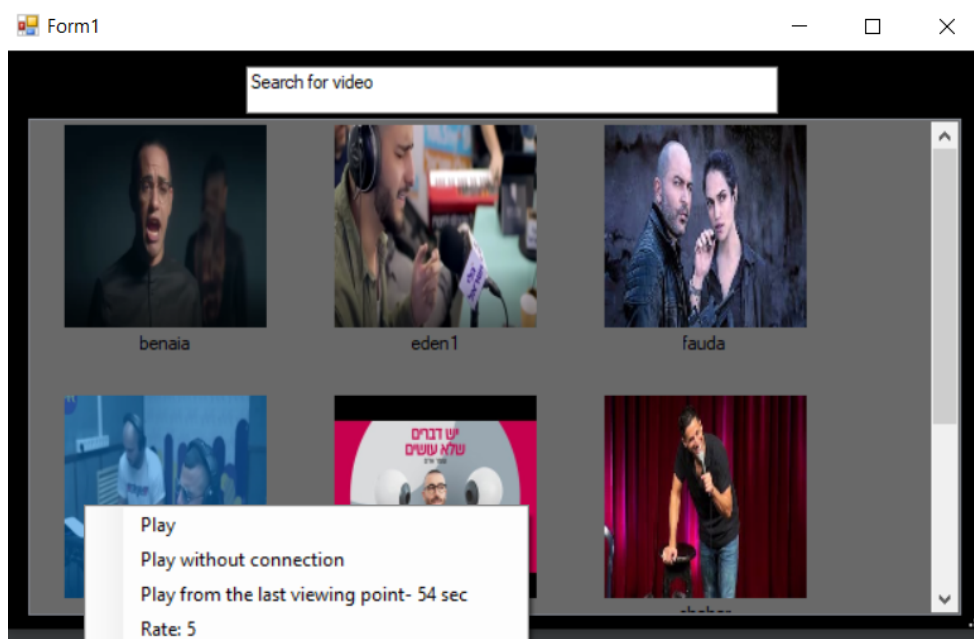
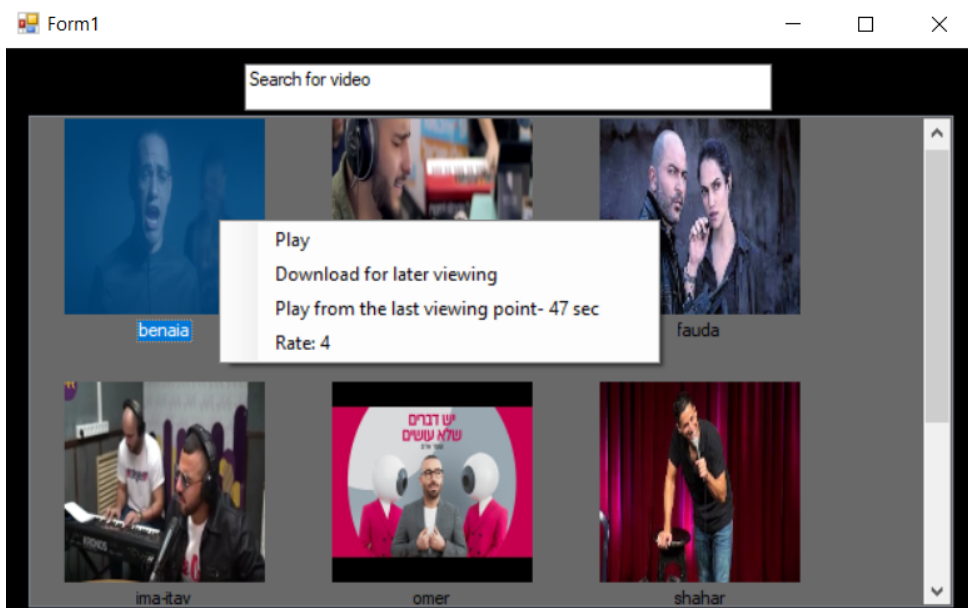
- לשנות בקבצים ,Rate\_Form.cs ,Login\_Form.cs ,Buttons\_Form.cs , Select\_Video\_Form.cs את כתובת המשתנה .pic\_directory.
- על מנת להפעיל התוכנה יש להריץ את processn ב#c# כאשר השרת כבר רץ.

### תיאור מסכים

**חלון מסך הפתיחה של המערכת מורכב משתי תיבות טקסט ומשני כפתורים.**  
תיבת הטקסט הראשונה מיועדת להכיל את שם המשתמש והשנייה את הסיסמא.  
הכפתור הראשון מיועד לכניסה למערכת, כלומר לבדוק אם המשתמש הכניס פרטים נכונים.  
הכפתור השני מיועד להרשמה למערכת, כלומר להוסיף משתמש חדש אם סיסמתו עומדת בתנאים הדרושים. חלון עם הערה המתאימה לבעיה יקפוץ כאשר המשתמש עשה משהו לא נכון – כמו להקליד שם משתמש או סיסמא לא נכונים, לנסות להירשם עם שם משתמש שכבר רשום או להקליד סיסמא שלא עומדת בדרישות המערכת. כאשר המשתמש הירשם או היכנס בהצלחה יפתח לו המסך הבא – מסך בחירת הסרטונים.



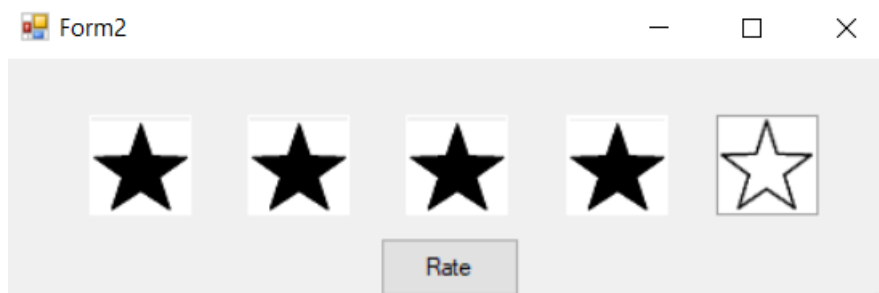
**חלון בחירת הסרטונים** של המערכת מורכב מתיבת טקסט ומרשימת תמונות. תיבת הטקסט מיועדת לספק חיפוש לפי שם הסרטון. רשימת התמונות מיועדת על מנת להציג למשתמש את הסרטונים שקיימים במערכת ומאפשרת לו לבחור במספר אפשרויות על כל סרטון. באמצעות לחיצה על המקש הימני תפתח רשימת אפשרויות. אם יבחר באפשרות הראשונה play- יפתח בפניו הסרט שמשודר בסטרימינג. אם יבחר באפשרות download for later viewing יפתח חלון הטעינה שיפורט בהמשך וירד הסרטון למחשב. אם יבחר באפשרות play from the last viewing point זה ינגן לו מנקודת הצפייה האחרונה שמצוינת. באפשרות האחרונה יופיע דירוג הסרטון. ניתן לראות בתמונה השנייה שאם הסרטון כבר הורד למחשב ניתנת אפשרות לצפות בו ללא חיבור – play without connection. כאשר נפתח הסרטון נפתח איתו חלון שליטה על הסרטון.



**חלון שליטה בסרטון וחלון הסרטון** של המערכת מורכבים 6 לחצנים חיוניים, 4 בחלון השליטה ו2 בחלון הסרטון. בחלון השליטה ישנו כפתור שגורם לסרטון לקפוץ 10 שניות קדימה במידה והמידע כבר הגיע ללקוח אם הוא בחר בנגינה בסטרימינג. מימינו יש כפתור שעוצר או ממשיך את הסרטון. מימינו יש כפתור שמשתיק את הסרטון. וכפתור סגירת חלון השליטה סוגר אותו ואת חלון הסרטון יחדיו ופותח את חלון בחירת הסרטונים. בחלון הסרטון יש את כפתור שינוי הגודל ואת כפתור המזעור. כפתור הX של חלון הסרטון לא פעיל. כאשר נגמר הסרטון יפתח יסגרו חלונות אלה ויפתח חלון של דירוג הסרטון.

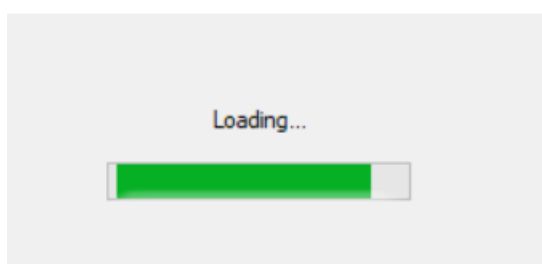


**חלון דירוג הסרטון** של המערכת מורכב מ-5 כפתורים המכילים תמונה וכפתור דירוג. כאשר המשתמש ילחץ על אחד מכפתורי התמונה יצבעו כל הלחצנים משמאלו בצבע שחור כולל הוא ומספר הכפתורים הצבועים בשחור הוא מספר הדירוג. כאשר הלקוח ילחץ על כפתור הדירוג, דירוגו יכנס למערכת וייסגר החלון ויפתח חלון בחירת הסרטונים.



**חלון הטעינה** של המערכת נפתח בשני מקרים – הראשון, כאשר המשתמש הוריד קובץ והקובץ עוד לא ירד. השני, כאשר המשתמש בחר לצפות מנקודת הצפייה האחרונה ונקודת צפייה זו עוד לא הגיע ללקוח (מכיוון שמשודר בסטרימינג).

Form3



# מדריך למפתח

## מבנה הקבצים

Name	Date modified	Type	Size
files	3/29/2020 8:07 PM	File folder	
songs	3/29/2020 8:07 PM	File folder	
WindowsFormsApplication8	3/29/2020 8:10 PM	File folder	
audio_handler.py	3/29/2020 7:58 PM	Python File	2 KB
client.py	3/29/2020 7:49 PM	Python File	24 KB
client_handler.py	3/29/2020 7:53 PM	Python File	12 KB
database.py	3/23/2020 6:19 PM	Python File	9 KB
encryption.py	3/23/2020 6:08 PM	Python File	5 KB
path_class.py	3/29/2020 7:01 PM	Python File	1 KB
server.py	3/23/2020 4:48 PM	Python File	3 KB
sub_process.py	3/29/2020 7:18 PM	Python File	1 KB

**Files** - תיקיה בלקוח שמכילה בהתחלה תמונות המשמשות ל GUI ונוספים לה תמונות המתארות את הסרטונים במערכת והסרטונים המורדים לצפייה ללא חיבור.

**Songs** - תיקיה בשרת שמכילה על כל סרטון במערכת את קבצי avi, wav, ותמונת bmp שלו.

**Windowsformapplication8** - תיקיה המכילה את קבצי gui ב#c#.

**Audio\_handler** - קובץ המכיל מחלקה משותפת שאחראית על ניגון האודיו.

**Client** - קובץ הלקוח של המערכת שאחראי על תקשורת עם השרת וה GUI.

**client\_handler** - קובץ המכיל מחלקה היורשת thread ומופעלת בשרת כל פעם שלקוח מתחבר.

**Database** - קובץ שאחראי על ניהול database קורא, וכותב לתוך מבנה הנתונים.

**Encryption** - קובץ המכיל מחלקות aes, rsa שדואגות להצפין את התעבורה ברשת.

**Path\_class** – מחלקה שממנה מקבלים את הכתובת של תיקיית הפרויקט

**Server** - השרת של המערכת דואג לפתוח לכל לקוח שמתחבר קליינט הנדלר שיטפל בו.

**Sub\_process** - סקריפט שמקצר את התהליך של הוספת סרטון למערכת. על מנת להוסיף

סרטון למערכת שיצרתי צריך שיהיה בתיקיה songs שבשרת את קבצי avi וה wav של

הסרטון ותמונת bmp שמתארת אותו. לשם כך יצרתי את הסקריפט הזה שמקצר את

התהליך בכך שצריך להכניס רק קובץ avi ותמונת bmp של הסרטון לתיקיה songs מבלי

לדאוג לקובץ wav. הסקריפט מקבל את שם קובץ avi של הסרטון שנמצא בתיקיה

songs ומפריד את האודיו שבקובץ avi לקובץ wav באמצעות תוכנה חיצונית הנקראת

ffmpeg. קובץ wav שנוצר נשמר תחת אותו שם של קובץ avi.

תיעוד על כל פונקציה ומחלקה נמצא בנספחים שאשלח בהם את הקוד של המערכת.

# רפלקציה

העבודה על הפרויקט הייתה עבורי עבודה מאתגרת, מעניינת ובעיקר מספקת ומהנה. במסגרת העבודה על הפרויקט למדתי הרבה מאוד דברים חדשים גם מבחינת ידע מקצועי וגם הרבה על עצמי, ועל איך שאני מתמודד עם קשיים ואתגרים חדשים. מכל שלב בפרויקט למדתי המון, בעיקר איך להתמודד כאשר אני מגיע למצב שהפתרון אינו ברור לי ואני צריך לחקור וללמוד הרבה דברים חדשים על מנת לפתור את זה בדרך הטובה והיעילה ביותר. מהפרויקט קיבלתי שיפור משמעותי בלמידה העצמאית שלי, בסבלנות שלי כאשר אני ניצב מול בעיה והרבה ידע חדש ששימש אותי במהלך בניית הפרויקט. הכלים שאספתי בזמן העבודה על הפרויקט אלו כלים שישמשו אותי להרבה דברים בתחום הטכנולוגי העתידי שאני מקווה לפתח, וכמובן לדברים נוספים בחיים. הכלים שאני בוחר לקחת איתי להמשך הם כלל הכלים שבהם השתמשתי בביצוע הפרויקט, כמו למידה באופן עצמאי, חקירה לעומק נושא שצריך לפתור, סבלנות ועבודה קשה עד להשגת המטרה והנאה מביצוע הפרויקטים שאעשה בעתיד.

הרבה קשיים ואתגרים ליוו אותי בהרבה שלבים בפרויקט, וכפי שצינתי קודם, למדתי מהם הרבה דברים חדשים שתרמו לי מאוד לידע המקצועי והרבה למידה על עצמי. הרבה פעמים כשמצאתי את עצמי במצב כזה, הבנתי שמה שעליי לעשות הוא לחשוב מחוץ לקופסא, כלומר מחוץ לידע שכבר יש לי, וללמוד דברים חדשים שיעזרו לי להגיע לפתרון היעיל ביותר. לדוגמא, בתחילת העבודה על הפרויקט כאשר הגעתי לחלק הסנכרון לא ידעתי איך להתמודד עם הבעיה מכיוון שאף פעם לא נתקלתי בכזו ולא היה שום דבר שעזר להגיע לפתרון באינטרנט. אחרי הרבה מאוד מזן של מחשבה וניסויים רבים הצלחתי להבין את הנושא והגעתי לתוצאה מיטבית. בנוסף לכך, הייתה לי תקלה במחשב עם ספריית pyaudio שהראתה לי שגיאה בניהול הזיכרון במחשב. כשאר ניסיתי להריץ על מחשב אחר הכל עבד ללא בעיה. אחרי הרבה מאוד התייעצות וחקירת הנושא הבנתי שזה נובע מבעיה בחומרת המחשב, כלומר הרמקולים והורדתי דרייבר עדכני שפתר את הבעיה. לו הייתי מתחיל היום את העבודה על הפרויקט, הייתי פועל אחרת בכך שהייתי מתכנן הרבה יותר את העבודה שעליי לעשות ומפרק אותה למשימות קטנות שעליי לעשות כל פעם. בנוסף לכך הייתי שומר כל חלק שעשיתי בפרויקט ורושם מה חלק זה עושה ואז הייתה לי אפשרות להשתמש בו בהמשך. הייתי מציב יעדים ברורים לכל שבוע על מנת לסיים כל חלק וככה להספיק הרבה יותר להתקדם. אם הייתי פועל אחרת מבחינת תכנון העבודה תהליך הביצוע היה הרבה יותר יעיל ומהיר. אני מאוד מרוצה מהמערכת שבניתי ושמה שבחרתי בנושא זה.

# ביבליוגרפיה

- [https://opencv-python-tutroals.readthedocs.io/en/latest/py\\_tutorials/py\\_gui/py\\_video\\_display/py\\_video\\_display.html](https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html) - איך לנגן קובץ avi באמצעות הספרייה cv2.
- <https://stackoverflow.com/questions/6951046/how-to-play-an-audiofile-with-pyaudio> - איך לנגן קובץ wav באמצעות הספרייה pyaudio.
- <https://www.pythonfortheclab.com/blog/using-pyzmq-for-inter-process-communication-part-1/> - מאמר על הממשק zmq.
- <https://stackoverflow.com/questions/30988033/sending-live-video-frame-over-network-in-python-opencv> - דרכי העברת פריימים ברשת.

# נספחים

הנספחים שארצה להגיש הם קבצי הקוד שכתבתי בשפת פייתון ו#c והם ישלחו בקובץ rar הנקרא TK\_Cinema\_Project.