

ספר פרויקט

נגן מוזיקה חכם

שם הפרויקט - socketify

שם המכללה - מכללת מקיף ה' אשקלון דרכא

סמל מוסד - 644450

שם סטודנט - ליאם ערבלי

ת"ז סטודנט - 214505653

שם המנחה - גל בר-און



תוכן עניינים:

2	תוכן עניינים:
4	הצעת פרוייקט
16	הצגת הפרויקט:
17	רקע
17	1. תיאור ורקע כללי:
17	2. תהליך המחקר
17	3. סקירת ספרות
18	VBR או CBR
18	מסירת וידאו באמצעות הורדת קבצים
19	מסירת וידאו באמצעות זרימה
19	Streaming (הזרמת מידע)
19	HTTP(Hypertext Transfer Protocol)
19	TCP/IP פרוטוקול בקרת השידור
20	WAV File(Waveform Audio File Format)
20	מטרות המערכת:
20	סקירת מצב קיים בשוק, אילו בעיות קימות:
21	מה הפרויקט אמור לחדש או לשפר
21	דרישות מערכת ופונקציונאליות
21	דרישות מערכת:
22	דרישות פונקציונאליות:
23	בעיות צפויות במהלך הפיתוח ופתרונות (תפעוליות, טכנולוגיות, עומס ועוד):
24	טכנולוגיות בשימוש:
25	שפות הפיתוח:
25	תיאור הארכיטקטורה הנבחרת:
26	חלוקה לתכניות ומודולים:
26	סביבת השרת (מקומי, וירטואלי, ענן, שירות אירוח)
26	ממשק המשתמש/לקוח - GUI
26	ממשקים למערכות אחרות:
26	שימוש במבני נתונים וארגון קבצים
26	מבני הנתונים:
27	מנגנוני התאוששות מנפילה/קריסה/תמיכה בטרנזקציות:
28	תרשימי מערכת מרכזיים
29	הפונקציות בנגן
29	תיאור המרכיב האלגוריתמי - חישובי
30	אלגוריתם סגמנטציה -
31	תיאור/התייחסות לנושאי אבטחת מידע
31	פתרונות לנושאי אבטחת מידע
32	משאבים הנדרשים לפרויקט:
33	תכנון הבדיקות שיבוצעו:
33	הגדרת הבעיה:
34	תהליכים עיקריים בפרויקט:

35.....	הפעלת המערכת:
38.....	קוד הפרויקט:
38.....	צד הלקוח – מבוסס WPF:
58.....	צד שרת
65.....	ביבליוגרפיה.

הצעת פרויקט הנדסאי תוכנה

סמל המוסד: 644450

שם מכללה: מקיף הדרכה

שם הסטודנט: ליאם ערבלני

ת"ז הסטודנט: 214505653

שם הפרויקט: Socketify

שם המנחה: גל בר-און



תיאור הפרוייקט:

אפליקציית נגן מוזיקה הדומה לspotify וapple music. מטרת המערכת היא לספק ללקוח חווית שמיעה של שירית המוזרמים (stream) דרך שרת ללא היכולת לשמור את השירים במערכת הלקוח באופן תמידי וללא פגיעה בחווית ההאזנה. קבצי wav יועברו מהשרת אל הלקוח במקטעים וישמרו בבאפר במערכת הלקוח באופן זמני על מנת למנוע העתקה של הקובץ באופן מוחלט. בזמן שהשיר יתנגן אצל הלקוח המערכת תדאג לשמור אצלו באפר של קובץ השיר של כX שניות קדימה Y שניות לאחר ותמחק כל מה שמעבר לתחום המוגדר על מנת למנוע העתקה ושמירה קבועה של הקובץ בשלמותו. בנוסף תהיה מערכת ניהול משתמשים דרכה כל משתמש יוכל ליצור ולשמור לעצמו במערכת רשימת שמע. למערכת תהיה גם מנהל מערכת ורק הוא יוכל להוסיף או למחוק שירים מהשרת ולקבל מידע אודות כל המשתמשים.

הגדרת הבעיה:

כאשר הלקוח רוצה להאזין לשיר קיימת האפשרות שהשיר יכיל תקיעות בשל בעיות חיבור זמניות לכן המערכת תשתמש בסגמנטציה (חלוקה דיגיטלית למקטעים) של קבצי wav ושמירתם בבאפר הדו כיווני בתחום המוגדר על מנת לאפשר ללקוח להאזין לשיר ולהריצו קדימה או אחורה ללא תקיעות הנגרמות מבעיות חיבור זמניות לרשת האינטרנט. יש גם למנוע מהמשתמש או מתוכנית אחרות אשר רצות במערכת ההפעלה של המשתמש להעתיק את קבצי האודיו. בנוסף יש להתמודד עם הפחתת ה latency של הזרמת המידע מהשרת אל הלקוח כדי למנוע עיכובים מיותר וחוויה תקועה.

את הבעיות האלו המערכת תפתור בעזרת האלגוריתמים הבאים:

• אלגוריתם לקידוד - WAV file

- הקידוד הרגיל של bitstream הוא פורמט LPCM (linear pulse-code modulation).
- קובץ WAV בנויים מכותרת (header), בלי אותה כותרת הקובץ לא יהיה תקין ולא יוכל לעבוד כראוי. הכותרת בנויה מ44 ביטים (bytes) וכל כמה ביטים ביחד מסמלים חלק בקובץ, והחלקים הם:
- בייטים 1-4: מסמן את הקובץ כקובץ riff, אורכו של כל תו הוא בייט אחד.
- בייטים 5-8: גודל הקובץ הכולל-8 בתים, בתים (מספר שלם של 32 סיביות). בדרך כלל, תמלא זאת לאחר היצירה.
- בייטים 9-12: כותרת סוג קובץ. לענייננו זה תמיד שווה ל "WAVE".
- בייטים 13-16: פורמט chunk marker כולל null נגרר.
- בייטים 17-20: אורך נתוני הפורמט כמפורט מעלה
- הקוד ממיר אותו למספר.

● אלגוריתם סגמנטציה-

- כל סגמנט יהיה בקפיצה של 2 שניות מהשיר המקורי.
- ה-socket שעליו הסגמנטים עוברים מהשרת ללקוח ייסגר בתנאי מסוים, בכדי לגרום לשירים לפעול אחד לאחר השני ללא המתנה מיותרת. כך גם אפשר לגרום לבאפר להפסיק את ההעברה באמצע במידה ולא רוצים להמשיך לשמוע את השיר המדובר.
- נרכיב מחדש את ה-header עבור כל סגמנט לקובץ WAV (יש כותרת המגדירה אותו).
- כל באפר יכול מספיק סגמנטים שיאפשרו ללקוח להריץ את השיר קדימה כמה עשרות שניות וכמה עשרות שניות לאחור בלבד (כ-20 30 שניות) וזאת על מנת למנוע את העתקת הקובץ.

● ניהול זיכרון במערכות הפעלה-

- מערכות הפעלה משתמשות במנגנונים שונים כדי לבודד תהליכים זה מזה ולמנוע מתהליך אחד לגשת לזיכרון, לקבצים ולנתונים של תהליך אחר. בידוד זה הכרחי לביטחון ויציבות המערכת.

מנגנון אחד המשמש לבידוד הוא זיכרון וירטואלי, המקצה לכל תהליך את מרחב הכתובות הפרטי שלו. המשמעות היא שכל תהליך יכול לגשת רק למיקומי הזיכרון שהוקצו לו ואינו יכול לגשת לזיכרון של תהליכים אחרים.

מנגנוני בקרת גישה משמשים גם להגבלת גישה לקבצים ולמשאבים אחרים. מנגנונים אלו מקצים הרשאות למשאבים, כגון קריאה או כתיבה, ומאפשרים רק לתהליכים בעלי ההרשאות הנדרשות לגשת למשאב. בסך הכל, בידוד התהליכים והגבלת הגישה למשאבים חשובים לאבטחה ויציבות של מערכות הפעלה מודרניות.

● תכנות אסינכרוני-

- תכנות אסינכרוני היא פרדיגמת תכנות המאפשרת לתוכנית לבצע מספר משימות במקביל. היא מאפשרת לתוכנית לבצע משימות ברקע תוך שהיא מאפשרת לתוכנית להמשיך לרוץ ולבצע משימות אחרות.

תכנות אסינכרוני שימושי למשימות שעשויות להימשך זמן רב להשלמתן, כגון בקשות רשת או פעולות קלט/פלט של קבצים, מכיוון שהיא מאפשרת לתוכנית להמשיך לבצע משימות אחרות במקום לחסום ולהמתין להשלמת המשימה האיטית.

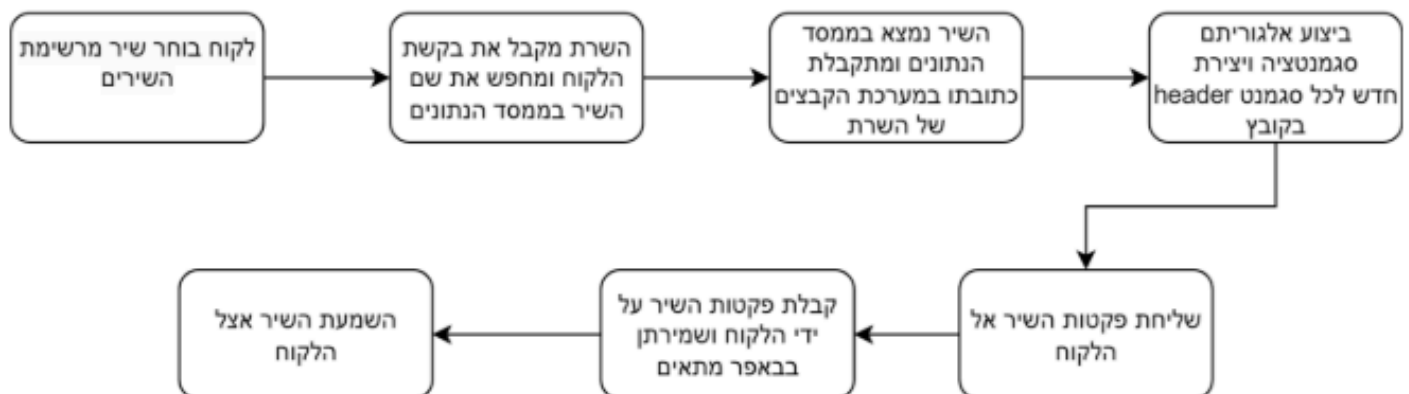
בתכנות אסינכרוני, משימה מופעלת בדרך כלל על ידי קריאה לפונקציה או מתודה, והתוכנית ממשיכה לבצע משימות אחרות בזמן שהמשימה היזומה פועלת ברקע. כאשר המשימה היזומה הושלמה, היא שולחת הודעה, הנקראת לעתים קרובות התקשרות חוזרת, לתוכנית, אשר תוכל להמשיך בביצוע המשימה הבאה.

באמצעות תכנות אסינכרוני אני אוכל להפחית את זמן ההשהייה (latency).

- שמירת קבצים בזכרון המטמון של השרת-

שמירת קבצים במטמון של שרת עשויה להפחית את זמן האחזור של המערכת על ידי מתן אפשרות לשרת לגשת לקבצים מהר יותר. מטמון הוא מיקום אחסון שמכיל נתונים שנכנסים אליו לעתים קרובות, כך שניתן לגשת אליו במהירות על ידי המערכת. כאשר קובץ נשמר במטמון, ניתן לאחזר אותו מהר יותר מאשר אם היה צריך לגשת אליו ממיקום אחסון איטי יותר, כגון כונן קשיח או אחסון רשת. זה יכול לעזור להפחית את משך הזמן שלוקח לגשת לקובץ ויכול לשפר את הביצועים הכוללים של המערכת.

ארכיטקטורת המערכת:



רקע תיאורטי בתחום הפרויקט:

הזרמת מידע (streaming)-

סטרימינג היא שיטה להעברת תוכן אודיו ווידאו דרך האינטרנט בזמן אמת, במקום לדרוש מהמשתמש להוריד ולאחסן את כל הקובץ במכשיר. זה מאפשר גישה מיידיית לתוכן, מה שהופך אותו לאפשרות נוחה עבור קבצים גדולים כגון סרטים או תוכניות טלוויזיה.

ישנם סוגים שונים של שירותי סטרימינג זמינים, כולל פלטפורמות הזרמת וידאו ושירותי הזרמת מוזיקה, שעשויים לדרוש דמי מנוי. איכות חווית הסטרימינג יכולה להיות מושפעת מהחוזק והיציבות של חיבור האינטרנט ורוחב הפס הזמין. חלק משירותי הסטרימינג מציעים את היכולת להתאים את איכות התוכן על סמך גורמים אלו.

בסך הכל, סטרימינג הוא דרך שימושית לגשת ולהינות מתוכן אודיו ווידאו באינטרנט ללא צורך באחסון פיזי. הוא מציע שיטה נוחה ויעילה לצריכת מדיה בעידן דיגיטלי.

פרוטוקול - http

HTTP (Hypertext Transfer Protocol) הוא מרכיב חיוני של האינטרנט המודרני, פרוטוקול HTTP ממלא תפקיד מכריע בהקלת התקשורת בין הלקוח (המכשיר של המשתמש) לבין השרת (שם מאוחסנים קבצי המוזיקה).

הנה איך פרוטוקול HTTP נכנס לפעולה בהקשר של אפליקציית הזרמת מוזיקה:

תקשורת שרת-לקוח:

כאשר משתמש פותח את אפליקציית הזרמת המוזיקה ומחפש שיר, מכשיר הלקוח שולח בקשת HTTP לשרת, תוך ציון הפעולה הרצויה (למשל, חיפוש שיר, בקשה לפלייליסט וכו').

שיטות בקשת HTTP:

פרוטוקול ה-HTTP מגדיר שיטות בקשה שונות, כגון GET, POST, PUT ו-DELETE. בהקשר של אפליקציית הזרמת מוזיקה, בקשת GET משמשת בדרך כלל בעת אחזור מידע על שירים או רשימות השמעה, בעוד שבקשת POST עשויה לשמש בעת העלאת נתוני משתמש או ביצוע שינויים ברשימות השמעה.

אחזור משאבים:

השרת מגיב לבקשת הלקוח בתגובת HTTP. תגובה זו עשויה לכלול מטא-דאטא על השיר המבוקש, כגון הכותרת, האמן, האלבום שלו וכתובת אתר המפנה לקובץ האודיו בפועל (במקרה זה, קובץ ה-WAV).

העברת קבצי אודיו:

לאחר מכן הלקוח שולח בקשות HTTP נוספות לשרת כדי לאחזר את קובץ האודיו בפועל. השרת מגיב עם קובץ האודיו המבוקש באמצעות פרוטוקול HTTP. זה נעשה בדרך כלל באמצעות בקשות של טווח בתים, מה שמאפשר ללקוח להוריד חלקים מקובץ האודיו בחתיכות.

STREAMING:

כשהלקוח מוריד את קובץ האודיו בחתיכות, הוא יכול להתחיל לנגן את השיר לפני הורדת הקובץ כולו. זה ידוע בשם סטרימינג. באמצעות התמיכה של פרוטוקול ה-HTTP בבקשות טווח, מתאפשר ללקוח לבקש חלקים ספציפיים מקובץ השמע, מה שמקל על השמעה חלקה.

אימות ואבטחה:

ניתן להשתמש בפרוטוקול HTTP גם לטיפול באימות משתמשים ולהבטחת תקשורת מאובטחת בין הלקוח לשרת. זה חשוב במיוחד באפליקציית הזרמת מוזיקה כדי להגן על נתוני המשתמש ולמנוע גישה לא מורשית לתוכן פרימיום.

IP/TCP:

חבילת פרוטוקולי TCP/IP היא קבוצה סטנדרטית של פרוטוקולי תקשורת המשמשים להעברת נתונים דרך האינטרנט ורשתות אחרות.

הרשת הבסיסית כמעט בכל היישומים מבוססי האינטרנט, כולל אפליקציות להזרמת מוזיקה.

הנה איך TCP/IP נכנס לפעולה בהקשר של אפליקציית הזרמת מוזיקה:

Transport Layer (TCP):

TCP הוא פרוטוקול מכון חיבור המבטיח מסירה אמינה ומסודרת של נתונים בין הלקוח לשרת. באפליקציית הזרמת מוזיקה:

כאשר לקוח יוזם חיבור לשרת (למשל, כדי לבקש שיר או רשימת השמעה), TCP יוצר חיבור אמין בין השניים.

מהימנות זו חיונית להזרמת נתוני אודיו מכיוון שהיא מבטיחה שכל חלקי הנתונים מגיעים שלמים ובסדר הנכון.

פילוח נתונים ובקרת זרימה:

TCP מפרק את נתוני האודיו למקטעים קטנים יותר לשידור. לכל קטע מוקצה מספר רצף. מנגנוני בקרת זרימה ב-TCP מונעים הצפה של הרשת או המכשיר המקבל על ידי התאמת קצב העברת הנתונים על סמך יכולתו של המקלט לטפל בו.

אישור ושידור חוזר:

TCP משתמש באישורים כדי לאשר את קבלת מקטעי הנתונים. אם קטע לא מקבל אישור בתוך מסגרת זמן מוגדרת, TCP מניח שהוא אבד ומשדר מחדש את הקטע. מנגנון אישור ושידור חוזר זה מבטיח מסירה אמינה של נתוני אודיו, וממזער את הסיכוי לאובדן מנות או השחתה.

ניהול חיבורים:

TCP מנהלת את הקמה, תחזוקה וסיום החיבורים בין הלקוח לשרת. זה חיוני כדי להבטיח ששני הקצוות מוכנים לשלוח ולקבל נתונים.

סטרימינג ותקשורת בזמן אמת:

באפליקציית הזרמת מוזיקה, תקשורת בזמן אמת חיונית להפעלה ללא הפרעה. TCP מספק זרם אמין ומסודר של נתונים, המאפשר ללקוח לקבל ולהשמיע את האודיו ברצף הנכון.

שכבת IP (פרוטוקול אינטרנט):

IP אחראי על ניתוב וכתובות בתוך הרשת. הוא מספק את הכתובת הדרושה כדי להבטיח שהנתונים מגיעים ליעד הנכון (כלומר, השרת המארח את קבצי המוזיקה).

שכבת רשת:

שכבת הרשת של מחסנית TCP/IP מטפלת בניתוב והעברה של מנות נתונים בין מכשירים על פני רשתות שונות. זה מאפשר תקשורת בין הלקוח לשרת, גם אם הם נמצאים ברשתות נפרדות.

:WAV file

קבצי WAV הם סוג של קבצי אודיו דיגיטליים המשמשים בדרך כלל לאחסון הקלטות קול במחשבים. הם ידועים באודיו האיכותי שלהם ובתמיכה הרחבה בקרב נגני מדיה ותוכנות עריכת אודיו. פורמט קובץ ה-WAV ממלא תפקיד משמעותי באחסון והעברת נתוני אודיו. הנה איך קובץ ה-WAV נכנס לפעולה בהקשר של אפליקציה כזו:

אחסון אודיו:

WAV הוא פורמט קובץ סטנדרטי לשמע לא דחוס. אפליקציות הזרמת מוזיקה עשויות לאחסן קבצי אודיו, כולל שירים ותוכן שמע אחר, בפורמט WAV בשרתים שלהם. אודיו לא דחוס מספק צליל באיכות גבוהה, שומר על הנאמנות המלאה של ההקלטה המקורית.

ייצוג אודיו:

כל קובץ WAV מכיל את דגימות האודיו הגולמיות, המייצגות את המשרעת של צורת גל האודיו בנקודות זמן שונות. ייצוג גולמי זה הופך אותו למתאים להשמעת אודיו בנאמנות גבוהה.

שידור משרת ללקוח:

כאשר משתמש מבקש שיר דרך אפליקציית הזרמת המוזיקה, השרת מגיב על ידי שליחת קובץ ה-WAV המתאים ללקוח. שידור זה מתרחש דרך הרשת, בדרך כלל באמצעות HTTP או פרוטוקול אחר.

:STREAMING

בזמן שידור קובץ ה-WAV, אפליקציית הזרמת המוזיקה עשויה להשתמש בטכניקות סטרימינג כדי לאפשר השמעה לפני הורדת הקובץ כולו. זה מושג לעתים קרובות על ידי שליחת נתוני אודיו בנתחים ומאפשרים ללקוח לנגן את המנות שהתקבלו כשהן מגיעות.

פענוח והשמעה:

התקן הלקוח מקבל את קובץ ה-WAV ומפענח אותו כדי לחלץ את נתוני האודיו. פענוח כולל המרת דגימות האודיו הגולמיות לפורמט שחומרת האודיו של המכשיר יכולה לנגן. בדרך כלל, אפליקציות סטרימינג משתמשות ב-codec אודיו לצורך דחיסה במהלך השידור ולאחר מכן מפענחות את הנתונים בצד הלקוח.

איכות שמע:

קובצי WAV לא דחוסים מציעים איכות שמע גבוהה, אך הם גדולים בגודלם. אפליקציות להזרמת מוזיקה עשויות להשתמש גם בפורמטי אודיו אחרים עם דחיסה (כגון MP3, AAC או Ogg Vorbis) לאחסון יעיל ושידור מהיר יותר. עם זאת, השמע המקורי והאיכותי עדיין עשוי להיות מאוחסן בפורמט WAV עבור מנויי פרימיום או עבור משתמשים שבוחרים ספציפית באפשרויות באיכות גבוהה יותר.

מטא-דאטא:

קובצי WAV יכולים לכלול מטא-דאטא, כגון מידע על האמן, האלבום, שם הרצועה ועוד. מטא-דאטא חיוני להצגת מידע על השיר בממשק הזרמת המוזיקה.

Latency:

latency, הידוע גם כזמן תגובה או עיכוב, מתייחס לזמן שלוקח לעיבוד הבקשה ולהחזרת תגובה במערכת מחשב או רשת. זהו מדד לכמה זמן לוקח למערכת להגיב לבקשה, והוא גורם חשוב שיכול להשפיע על הביצועים וחווית המשתמש של המערכת.

במערכות מחשב, latency יכול להיות מושפע מהמהירות ומהיכולות של רכיבי החומרה והתוכנה של המערכת, ממורכבות הבקשה ומעומס העבודה של המערכת. ברשתות, latency מושפע מהמרחק בין המכשירים, מהירות החיבור וכמות התעבורה ברשת.

latency נמדדת בדרך כלל באלפיות שניות (ms) ומהווה שיקול חשוב ביישומים בזמן אמת, כגון משחקים מקוונים ומסחר פיננסי, שבהם זמני תגובה מהירים הם קריטיים. בדרך כלל מועדף latency נמוך יותר מכיוון שהוא מצביע על כך שבקשות מעובדות ותגובות מוחזרות מהר יותר. עם זאת, במקרים מסוימים, latency גבוה יותר עשוי להיות מקובל אם הוא מאוזן על ידי גורמים אחרים, כגון אבטחה מוגברת או אמינות.

הבנה ואופטימיזציה של latency חשובים בתכנון והערכה של מערכות מחשב. על מנת לשפר את הביצועים וחווית המשתמש של מערכת, חשוב לזהות ולטפל בכל גורם שעלול לתרום לlatency גבוה. זה עשוי להיות כרוך באופטימיזציה של רכיבי החומרה והתוכנה, שיפור חיבורי הרשת והפחתת עומסי העבודה כדי להבטיח שהבקשות יעובדו בצורה יעילה ואפקטיבית.

סגמנטציה-

סגמנטציה היא טכניקה בשימוש נרחב בהעברת נתונים הכוללת חלוקת כמות גדולה של נתונים ליחידות או מקטעים קטנים יותר לפני שידורם ברשת. לתהליך זה מספר יתרונות, כולל יעילות משופרת, אמינות וטיפול טוב יותר בנתונים על ידי התקני רשת. הנה איך סגמנטציה נכנסת לפעולה:

עיבוד נתונים לסטרימינג:

קבצי אודיו, במיוחד לא דחוסים כמו WAV, יכולים להיות גדולים. פילוח כרוך בפירוק קבצים אלה לחתיכות או מנות קטנות יותר וניתנות לניהול. זה חיוני לסטרימינג, ומאפשר ללקוח להתחיל לנגן את האודיו בזמן ששאר הקובץ עדיין משודר.

בקשות לטווח HTTP:

פילוח משמש לעתים קרובות בשילוב עם בקשות טווח HTTP. כאשר הלקוח מבקש חלק מסוים מקובץ האודיו, הוא יכול להשתמש בכותרות טווח HTTP כדי לציין את הטווח הרצוי של בתים. לאחר מכן השרת מגיב על ידי שליחת החלק המבוקש בלבד, תוך מזעור העברת נתונים מיותרת.

הפחתת latency:

על ידי שימוש בפילוח, ההשהיה בהתחלת השמעת אודיו מצטמצמת. הלקוח יכול לקבל ולהשמיע במהירות את הנתונים הראשוניים של קובץ השמע מבלי לחכות להורדת הקובץ כולו.

הזרמת קצב סיביות אדפטיבית:

פילוח הוא קריטי גם בהזרמת קצב סיביות אדפטיבית, שבה גרסאות שונות של קובץ השמע זמינות בקצבי סיביות משתנים. הלקוח יכול לעבור באופן דינמי בין גרסאות אלו בהתבסס על תנאי הרשת. פילוח מאפשר ללקוח לבקש ולעבור לקצב סיביות אחר בצורה חלקה.

שחזור שגיאות (error recovery):

במקרה של בעיות רשת או אובדן מנות, פילוח מאפשר שידור חוזר ממוקד של נתחים ספציפיים במקום שידור מחדש של הקובץ כולו. זה משפר את האמינות הכוללת של תהליך הסטרימינג.

Buffering:

הלקוח שומר בדרך כלל על מאגר לאחסון נתחי שמע שהתקבלו. מאגר זה מבטיח זרם רציף של נתוני אודיו, גם אם יש תנודות במהירות הרשת. פילוח מאפשר מילוי יעיל של מאגר זה על ידי כך שהוא מאפשר ללקוח לבקש ולקבל נתחים על סמך מצב המאגר שלו.

יעילות משאבים:

פילוח עוזר לייעל את השימוש במשאבים הן בצד הלקוח והן בצד השרת. לקוחות מורידים רק את החלקים הדרושים להפעלה מיידיית, ושרתים משדרים נתחים קטנים יותר, ומפחיתים את העומס בשני הקצוות.

חיפוש בתוך מסלולים:

פילוח מאפשר למשתמשים לחפש חלקים שונים של רצועה מבלי להוריד את כל הקובץ. הלקוח יכול לבקש מקטעים ספציפיים המתאימים למיקום ההשמעה הרצוי.

בסך הכל, סגמנטציה הוא היבט חשוב של העברת נתונים הממלא תפקיד מכריע באופטימיזציה של השידור והטיפול בנתונים ברשתות. טכניקה זו נמצאת בשימוש נרחב בתעשיות שונות והוכחה כיעילה בשיפור היעילות והאמינות של העברת הנתונים.

פרוטוקול RTMP-

RTMP הוא פרוטוקול בעל ביצועים גבוהים להזרמת אודיו, וידאו ונתונים אחרים דרך האינטרנט. הוא משמש פלטפורמות ושירותי סטרימינג רבים כדי לספק תוכן חי ולפי דרישה. RTMP הוא פרוטוקול מבוסס TCP המאפשר תקשורת עם אחזור נמוך ושימוש יעיל ברוחב הפס. ניתן להשתמש בו כדי להזרים אודיו ווידאו במגוון פורמטים, כולל H.264, HEVC ו-9VP.

RTMP פועל על ידי יצירת חיבור בין השרת ללקוח באמצעות TCP. לאחר יצירת החיבור, השרת והלקוח יכולים להחליף נתונים בזמן אמת. RTMP תומך במספר סוגי הודעות שונים, כולל נתוני אודיו ווידאו, metadata והודעות בקרה.

RTMP משתמש במספר טכניקות כדי להבטיח שימוש יעיל ברוחב פס והשהייה נמוכה. לדוגמה, הוא יכול להשתמש בשחזור אובדן מנות כדי לשחזר מנות שאבדו, והוא יכול להתאים באופן דינמי את איכות הזרם בהתבסס על רוחב הפס הזמין. RTMP תומך גם במיתוג זרמים, המאפשר ללקוח לעבור בצורה חלקה בין זרמים שונים בהתבסס על תנאי הרשת.

פרוטוקול HLS-

HLS הוא תקן פתוח שפותח על ידי אפל להזרמת אודיו ווידאו דרך האינטרנט. זה נתמך באופן נרחב וניתן להפעיל אותו בכל מכשיר עם נגן תואם HLS, כולל סמארטפונים, טאבלטים וטלוויזיות חכמות. HLS פועל על ידי חלוקת הזרם הכולל לסדרה של נתחים קטנים יותר, המועברים ללקוח באמצעות פרוטוקול HTTP. זה מאפשר אספקת HLS באמצעות שרתי אינטרנט סטנדרטיים, ללא צורך בשרתי סטרימינג מיוחדים.

הנה איך HLS נכנס לפעולה בהקשר של אפליקציית הזרמת מוזיקה:

הזרמת קצב סיביות אדפטיבית:

אחת התכונות המרכזיות של HLS היא הזרמת קצב סיביות אדפטיבית. HLS מחלק את תוכן המדיה למקטעים קטנים בגודל שווה ומספק גרסאות מרובות של כל פלח בקצבי סיביות שונים. מכשיר הלקוח עובר באופן דינמי בין גרסאות אלו בהתבסס על תנאי הרשת שלו, מה שמבטיח חווית סטרימינג חלקה גם במהירויות רשת משתנות.

תאימות:

HLS נתמך במגוון רחב של מכשירים ופלטפורמות, כולל מכשירי iOS, מכשירי אנדרואיד, דפדפני אינטרנט, טלוויזיות חכמות ועוד. תאימות רחבה זו הופכת את HLS לבחירה מתאימה להגיע לבסיס משתמשים מגוון עם סוגים שונים של מכשירים.

Segmentation:

בדומה לפרוטוקולי סטרימינג אחרים, HLS מפלח את תוכן המדיה לנתחים או פלחים קטנים. קטעים אלה הם בדרך כלל באורך של כמה שניות. פילוח מאפשר הזרמה יעילה, חציצה והתאמה לתנאי הרשת המשתנים.

מבוסס HTTP:

HLS ממנפת פרוטוקולי HTTP סטנדרטיים לתקשורת. זה הופך אותו לידידותי לחומת האש ומאפשר לו לנצל תשתית קיימת להעברת תוכן. הוא פועל על יציאות HTTP סטנדרטיות (80 עבור חיבורים לא מאובטחים ו-443 עבור חיבורים מאובטחים), מה שמפשט את תצורת הרשת.

רשימת השמעה בצד הלקוח:

HLS משתמש בקובץ רשימת השמעה, המכונה רשימת ההשמעה M3U8, כדי לציין את הרצף וכתובות האתרים של מקטעי המדיה. הלקוח מוריד את קובץ הפלייליסט ומשתמש בו כדי לבקש ולהוריד את הקטעים המתאימים. גישה זו מאפשרת גמישות בהעברת תוכן ובמניפולציה של רשימות השמעה בצד הלקוח.

השהה, חפש והמשך:

HLS תומך בתכונות כמו השהייה, חיפוש וחיידוש השמעה. משתמשים יכולים להשהות זרם, לחפש למיקום אחר ולחדש את ההשמעה בצורה חלקה. הלקוח מבקש מקטעים ספציפיים המתאימים למיקום ההשמעה הרצוי.

הצפנה ואבטחה:

HLS תומך בהצפנת תוכן באמצעות פרוטוקולים כמו HTTPS ותומך בניהול זכויות דיגיטליות (DRM) להגנה על תוכן המוגן בזכויות יוצרים. זה מבטיח אספקה מאובטחת והשמעה של תוכן פרימיום או מוגן.

בקרת אחזור:

HLS בדרך כלל מציגה רמה מסוימת של חביון בשל אופי הסטרימינג המבוסס על פלחים. עם זאת, ניתן להשתמש בטכניקות כמו צמצום משך הקטע או שימוש בגרסאות HLS עם אחזור נמוך כדי למזער את זמן האחזור עבור תרחישי סטרימינג בשידור חי.

HLS הוא פרוטוקול סטרימינג מאומץ נרחב המספק הזרמת קצב סיביות אדפטיבית, תאימות רחבה של מכשירים וקלות אינטגרציה עם תשתית קיימת. הוא מתאים לאספקת תוכן אודיו באיכות גבוהה באפליקציית הזרמת מוזיקה, ומציע תכונות כמו סטרימינג אדפטיבי, אבטחה והשמעה חלקה על פני מכשירים ותנאי רשת שונים.

במערכת שאני אבנה אני אוכל להשתמש בפרוטוקול HLS או RTMP וללמוד אותם ולממש את שיטת העבודה שלהם באמצעות פרוטוקול אשר אני אפתח בעצמי.

שפת תכנות ממסד נתונים: SQL

צד שרת:

צד השרת יהיה אחראי על ניהול המשתמשים, חלוקת הקבצים למקטעים ושליחתם ללקוחות.

שפת תכנות בצד שרת: python

צד לקוח:

צד לקוח ישמש להצגת ממשק משתמש המערכת.

אך ראשית יש לבצע הזדהות בכניסה למערכת.

הפעולות אשר יוכל המשתמש לבצע הן:

- יצירת רשימת שירים, הוספת שירים לרשימה והסרתם.
- השמעת שירים מהשרת מרשימת שירים של המשתמש.
- חיפוש שירים במאגר הנתונים של השרת.
- השמעת שירים ממאגר הנתונים.
- פעולות אשר רק מנהל המערכת יוכל לבצע:
- הצגת המידע של כל המשתמשים.
- הוספה או הסרה של שירים מהמערכת.
- עריכת נתוני המשתמשים.

שפת תכנות צד לקוח:

JAVA

פרוטוקולי תקשורת:

http, tcp/ip

לוחות זמנים:

תאריך סיום השלב	שלבי עבודה
31.12.23	שליחת הצעת פרוייקט
2.1.24	חקירה ולמידה אודות הנושא לעומק
20.1.24	הקמת צד שרת
21.1.24	הקמת צד לקוח בסיסי
15.2.24	פיתוח אלגוריתם סגמנטציה
17.2.24	חיבור בין צד שרת ללקוח
17.2.24	בדיקת המערכת בפעם הראשונה
20.2.24	הסקת מסקנות וביצוע שיפורים
7.3.24	פיתוח gui מתקדם
8.3.24	שיפור שדרוג וייעול המערכת
8.3.24	בדיקת המערכת בפעם השניה
22.3.24	הסקת מסקנות, תיקון ושיפור הבעיות
25.5.24	הכנת המערכת להגשה

חתימת הסטודנט:

הצגת הפרויקט:

הפרויקט יעסוק בנגן חכם

פרויקט זה בא לענות על הצורך להשמעה מהירה של שירים, ללא המתנה להורדה מלאה של השיר שאותו רוצים לשמוע, אלא הורדה והשמעה בו זמנית בעזרת נגן קבצי WAV ובאפר.

הפרויקט יעסוק במערכת המזרימה שירים (במקרה שלנו wav files) מהשרת ללקוח בעזרת buffer, משמעות הדבר היא שלפני שמסתיימת הזרמה של שיר במלואו למחשבו של הלקוח, השיר מתחיל להתנגן, ובו זמנית המשך השיר מוזרם למחשב (כמו באפליקציה YOUTUBE).

המערכת תאפשר לבחור שיר מספריית שירים המוטמעת בלקוח, או להפעיל פלייליסט (playlist). משמע כל השירים מן הספרייה ישמעו אחד לאחר השני. כל זה נעשה בעזרת חיתוך השיר לחלקים קטנים, המרת החלקים מbytes string ו העברתם בעזרת socket מהשרת ללקוח, המרתם חזרה string והשמעה של הקובץ בצד הלקוח באופן רציף, ללא המתנה להורדה של החלק הבא או המתנה להורדה של כל השיר. דבר זה יעיל ו חוסך את זמן ההמתנה.

המערכת תאפשר גם לנגן סרטים וקבצי אודיו אשר במחשב הלקוח.

הגדרת הבעיה

המערכת תנהל את דרך ההורדה של השירים ותאפשר הורדה של מספר קבצי אודיו בו-זמנית ע"י שימוש בסגמנטציה (חלוקה דיגיטלית למקטעים) של קבצי ה WAV. דבר זה יכול למנוע מצבים של קטיעה בשירים ואף לאפשר האזנה או צפייה לקובץ ללא תלות בחיבור אינטרנטי. אזי המערכת תצטרך להתמודד עם הורדה של מידע באופן חד ערכי (כאשר כל פעם יורד חלק אחד מהמידע / שיר) להעברת מידע בצורה מקבילית (ביטים אשר יורדים במקביל).

רקע

1. תיאור ורקע כללי:

הפרויקט הינו נגן אשר מטרתו להטעין שירים בצורה מהירה כך שגם בזמן שאין חיבור אינטרנטי ניתן עדין לשמוע את המוזיקה או לראות את הסרט ואף לייצר רשימת השמעה (פלייליסט)

פרויקט זה בא לענות על הצורך להשמעה מהירה של שירים, ללא המתנה להורדה מלאה של השיר שאותו רוצים לשמוע, אלא הורדה והשמעה בו זמנית בעזרת נגן קבצי WAV ובאפר.

2. תהליך המחקר

בתחילת דרכי בפרויקט חיפשתי מקורות מידע אשר יכולים לעזור לי לחקור את הנושא, ומצאתי מגוון של מאמרים מקצועיים על הזרמת מידע, כמו כן חיפשתי הסברים ללימוד עצמי על WPF וקבצי WAVE על מנת להבין כיצד הדברים עובדים ולהשתמש בכלים ש קיימים בצורה הטובה והיעילה ביותר.

בנוסף המנחה גל בראון סייע לי בכך ששלח לי מידע ומאמרים בנושאים השונים בכדי לשפר ולסייע לפרויקט גמר בכדי להגיע ל פרויקט הכי טוב במסגרת הזמן שהיה נתון.

3. סקירת ספרות

וידאו היה מדיה חשובה לתקשורת ובידור במשך עשורים רבים, בהתחלה וידאו נתפס ושודר בצורה אנלוגית.

תקשורת בוויידאו בעל רשת באיכות טובה זה מסובך ע"י מספר גורמים ביניהם זמן וידוי של רוחב פס לא ידוע, עיכוב והפסדים. וכמובן בעיות ייחודיות כמו איך לחלק את משאבי הרשת בין זרמי מידע רבים בצורה יעילה .

קיים מגוון רחב של תקשורת בוויידאו וגם אפליקציות סטרימינג, שיש להם תנאי הפעלה שונים מאוד או תכונות. לדוגמא תקשורת באפליקציית ווידאו עשויה להיות עבור תצורת תקשורת point to point או עבור תצורת תקשורת multicast או broadcast, והוויידאו יתכן ויהיה מקודד מראש (מאוחסן) או

יתכן ומקודד בזמן אמת (לדוגמא: טלפון וידאו אינטראקטיבי או קיום שיחות ועידה בוויידאו).

תצורות התקשורת בוויידאו הן: broadcast, multicast, Point-to-point

תצורת התקשורת הפופולארית ביותר היא broadcast, היא תצורת תקשורת מאוד יעילה עבור שליחת תכנים למספר משתמשים באותו הזמן, וגם המערכת חייבת להיות מעוצבת כך שהיא יכולה

לספק כל קבלה עתידית עם האות הנדרש. זאת בעיה חשובה, מאז שנמענים שונים עשויים לחוות מאפייני ערוצים שונים, וכתוצאה המערכת מעוצבת לעיתים קרובות עבור הערוץ הגרוע ביותר.

Point-to-point מאפיין חשוב בתצורה זו הוא שתמיד יש ערוץ אחורי בין המקבל ל שולח. אם הערוץ האחורי קיים המקבל יוכל לספק תגובה לשולח אשר השולח יכול להשתמש בה לאחר מכן

כדי להתאים את העיבוד שלו . IP-Multicast לדוגמא, broadcast יחיד ל רבים אבל שונה מ Multicast- קידוד וידאו בזמן אמת לעומת קידוד וידאו מראש - וידאו עשוי להתקבל ומקודד לתקשורת בזמן אמת או עשוי להיות מקודד מראש ומאוחסן לצפייה מאוחרת יותר. אפליקציות אינטראקטיביות הן דוגמא אחת לאפליקציות אשר דורשות קידוד בזמן אמת, לדוגמא שיחת ועידה.

מערכת תקשורת וידאו מעוצבת באופן משמעותי אם המאפיינים של ערוץ התקשורת כגון רוחב פס, עיכוב, איבוד, הם סטטיים או דינמיים

VBR או CBR

חלק מן הערוצים תומכים ב CBR לדוגמא ISDN, וחלק מן הערוצים תומכים ב VBR לדוגמא אחסון DVD, ותקשורת דרך רשת משותפת. לרצף של וידאו יש בדרך כלל מורכבות של משתנה בזמן.

לכן קידוד וידאו על מנת להשיג קבוע ויזואלי איכותי נדרש משתנה דירוג ביטים וקידוד עבור קבוע דירוג הביטים ש יפיק זמן וידוי איכותי. ברור שזה דבר חשוב לתאם א דירוג הביטים של הוידאו למה ש הערוץ יכול לתמוך. על מנת להשיג את ה חוצץ המשמש בדרך כלל כדי לזוג את המקודד הוידאו לערוץ ומכניזם המספק משוב מבוסס על החוצץ מלאות לווסת את הגסות/ עדינות של הכימות ובכך את החלק וידאו.

ברשתות מנות מוחלפות כמו אתרנט LAN והאינטרנט, הם רשתות משותפות איפה ש החבילות האישיות של המידע עשויות להציג משתני עיכוב, העשויים להגיע מחוץ להזמנה, או להיאבד לחלוטין.

לחלופין, רשתות מעגל מוחלף, כמו רשת טלפונים מוחלפת ציבורית (PSTN) או ISDN, המשאבים השמורים והמידע בעל העיכוב המתוקן, מגיע בהזמנה, למרות זאת המידע עדיין יכול להיות פגום על ידי שגיאות סיביות או שגיאות פרץ.

מסירת וידאו באמצעות הורדת קבצים

כנראה הגישה הפשוטה ביותר למסירת וידאו של האינטרנט הוא על ידי משהו דומה להורדת קובץ, אבל אנחנו מתייחסים אליו כמו וידאו הורד כדי לזכור שזה וידאו ולא קובץ כללי.

באופן ספציפי, הורדת וידאו דומה להורדת קובץ, אך היא גדולה קובץ. גישה זו מאפשרת שימוש במנגנוני מסירה מבוססים, עבור דוגמה ל- TCP כשכבת התעבורה או כ- HTTP FTP בשכבות הגבוהות יותר.

מסירת וידאו באמצעות זרימה

מסירת וידאו על ידי הזרמת וידאו מנסה להתגבר על הבעיות המשויכות להורדת קבצים, וגם מספק כמות משמעותית של יכולות נוספות. הרעיון הבסיסי של הזרמת וידאו הוא לפצל את הווידאו לחלקים, לשדר חלקים אלה ברצף ולאפשר למקלט לפענח ולהשמיע את הווידאו כאשר חלקים אלה מתקבלים, ללא צורך בלהמתין עד שהסרטון כולו יימסר.

Streaming (הזרמת מידע)

טכנולוגיית אינטרנט להעברת מדיה דיגיטלית למשתמש קצה על ידי ספק תוכן באופן מתמשך ורציף.

התוכן מועבר באופן שוטף מהרגע בו החל המשתמש לצרוך אותו. המונח מתייחס בדרך כלל להעברת תוכן של רשתות תקשורת ותוכן מתמשך לדוגמא שידורי רדיו או וידאו.

בטרם התקבל כל המידע המרכיב אותו, שכן שידור מידע בסטרימינג מאפשר האזנה או צפייה בקובץ (למשל פריימים בודדים כל פעם, במקרה של וידאו) בחלקים עוקבים קטנים "מוזרמת" המדיה.

הטכנולוגיה מביאה לחסכון בכמות התעבורה שעוברת ברשת אל מול איכות הווידאו המוצג, ומאפשרת צפייה ישירה באינטרנט או צפייה מקוונת בסרטים שידורים חיים.

Hypertext Transfer Protocol (HTTP)

פרוטוקול תקשורת שרת-לקוח המבוסס טקסט הנועד להעברת דפי HTML ואובייקטים שהם מכילים (כמו תמונות, קובצי קול, סרטוני פלאש וכו') ברשת.

פרוטוקול זה פועל בשכבת היישום (7) של מודל OSI ובשכבת היישום של מודל TCP/IP.

TCP/IP פרוטוקול בקרת השידור

(TCP) הוא אחד הפרוטוקולים העיקריים של חבילת פרוטוקול האינטרנט. מקורו ביישום הרשת הראשוני בו הוא השלים את פרוטוקול האינטרנט (IP). לכן מכונה בדרך כלל החבילה כולה TCP/IP.

TCP מספקת משלוח אמין, מסודר ובודק שגיאות של זרם שמיניות (בתים) בין יישומים הפועלים על מארחים המתקשרים דרך רשת IP. יישומי אינטרנט מרכזיים כמו האינטרנט דורשים את האמינות והסדר שמספק TCP כדי להבטיח שהנתונים המועברים יגיעו ליעדם ללא שגיאות.

IP מספק את הכתובת של השרת ומוודא שהאתר ישלח לכתובת הייעודית שלו.

WAV File(Waveform Audio File Format)

הוא קובץ שמע, שפותח על ידי IBM ו- MICROSOFT לאחסון audio bitstream במחשב WAV File משתמשת בפורמט RIFF לאחסון מידע בחלקים (chunks)

זהו הפורמט העיקרי המשמש במערכות Microsoft Windows עבור שמע גולמי ולא דחוס בדרך כלל, ומפני שהשמע גולמי ולא דחוס הוא תופס יותר מקום מאשר קבצי שמע אחרים (עדכניים יותר) במחשב. גודלה של דקה ב-WAV אמורה לשקול כ-10 MB

גודלו המרבי של קובץ זה יכול להגיע עד 4GB

מטרות המערכת:

פרויקט זה עוסק במערכת המזרימה שירים (במקרה שלנו מדובר ב wav files) משרת ללקוח בעזרת buffer, משמעות הדבר היא שלפני שמסתיימת הזרמה של שיר במלואו למחשבו של הלקוח, השיר מתחיל להתנגן, ובו זמנית המשך השיר מוזרם למחשב (כמו באפליקציה YOUTUBE). ניתן לבחור שיר מספריית שירים המוטמעת בלקוח, או להפעיל פלייליסט (playlist), משמע כל השירים מן הספרייה ישמעו אחד לאחר השני. כל זה נעשה בעזרת חיתוך השיר לחלקים קטנים, המרת החלקים מbytes string, העברתם בעזרת socket מהשרת ללקוח, המרתם חזרה לstring והשמעה של הקובץ בצד הלקוח באופן רציף, ללא המתנה להורדה של החלק הבא או המתנה להורדה של כל השיר. דבר זה יעיל וחוסך בזמן המתנה מיותר.

מטרה משנית של המערכת היא לתת אופציה לניגון סרטים וקבצי אודיו אשר במחשב הלקוח.

סקירת מצב קיים בשוק, אילו בעיות קימות:

אחד האפליקציות הכי מתקדמות בנושא היא אפליקציית YouTube שבה פותחה אופציה חדשה שבה ניתן לנגן שיר/סרט במלואו בזמן שאין חיבור לאינטרנט וגם כאשר מכשיר הפלאפון סגור.

השירות הינו שירות חדש ובתשלום שזהו חסרונם של השירות.

תוכנה נוספת היא Spotify, מערכת שמזרימה מוזיקה (בתשלום) במערכת זו אם יש מינוי Premium אפשר להוריד את השירים, האלבומים, הפלייליסטים והפודקסטים כך שאפשר להאזין להם ללא חיבור אינטרנט.

החסרונות של מערכת זו הם שהיא מתעסקת בעיקר בהזרמת מוזיקה ולא בוידאו ובנוסף היא בתשלום.

תוכנה נוספת שמשתמשת בטכנולוגיה של הורדת מדיה היא Netflix, כאשר נרצה לראות סרט או סדרה ללא חיבור אינטרנט נוכל להוריד מבעוד מועד את הקובץ דרך המערכת של Netflix ונוכל לראות את אותו סדרה / סרט בזמן ללא חיבור אינטרנט. ההבדל בין המערכת של Netflix למערכת שפיתחתי הינה שהם משתמשים במערכת רק עבור סרטים, סדרות וקבצי אודיו והם לא משתמשים בטכנולוגיה על מנת להזרים מוזיקה. הטכנולוגיה היא דומה אך משתמשים בה למטרה שונה.

שירותים נוספים שמשתמשים בטכנולוגיה דומה הינם שירותי הסטרימינג של yes ושל hot. ההבדל שהם לא משתמשים בקו אינטרנטי אך ניתן לראות סדרות וסרטים שיושבים על המערכת ללא תלות בשידורי הטלוויזיה. הטכנולוגיה שלהם שונה אך הרעיון הבסיסי הוא דומה - צפייה בסרטים ללא תלות בחיבור.

נראה שישנם הרבה אפליקציות שעוסקות ומתחרות בתחום הסטרימינג, השוק הינו תחרותי מאוד, אך אחד החסרונות של המערכות שרובם הם בתשלום ולא מוצעות בחינם. נראה כי על מהירות ויעילות צריך לשלם.

בנוסף נראה כי ניתן לראות את תחום הסטרימינג מופיע בתחומים רבים ולא דווקא בנגנים, עצם פיתוח הטכנולוגיה שמייעלת מהירות העברת מידע יכול להתפרש לתחומים רבים ולא דווקא מוזיקה או וידאו (לדוגמה יעילות מערכות הפעלה).

אומנם קיימות אפליקציות ותוכנות רבות בתחום אך הפרויקט שלי אמור לשנות את דרך ההורדה של השירים מהורדה חד פעמית של קובץ אודיו יחיד להורדה של מס' קבצים בו זמנית ע"י שימוש בסגמנטציה (חלוקה דיגיטלית למקטעים) של קבצי WAVE.

מה הפרויקט אמור לחדש או לשפר

כיום, הורדת שירים נעשית לרוב באמצעות הורדה חד פעמית של קובץ אודיו יחיד. גישה זו עלולה לגרום למספר בעיות, כגון קטיעה בשירים ותלות בחיבור אינטרנטי. הפרויקט מציע פתרון חדש להורדת שירים, המאפשר הורדה מקבילה של מספר קבצים בו זמנית. פתרון זה מבוסס על שימוש בסגמנטציה (חלוקה דיגיטלית) של קבצי WAV למקטעים.

דרישות מערכת ופונקציונאליות

דרישות מערכת:

- יכולה לבצע פונקציה אחת או יותר.
- יש לה ממשק אחד דרכם יכולים ישויות חיצוניות ליזום את ביצוע הפונקציות ואת קבלת השירותים.
- יכולת ליצור תקשורת למערכת נוספת.

- המערכת נדרשת לספק למשתמש ממשק נוח לשימוש בו הוא יכול להפעיל את השירות, להפסיק אותו להריץ קדימה ואחורה, להגביר ולהנמיך רמקולים ולעצור.
- יחד עם זאת המערכת נדרשת לסיים להוריד באופן מהיר ויעיל את הקבצים כך שבזמן של הפסקת אינטרנט הוא עדין יוכל ליהנות מהשימוש במערכת.

- ניתן לבחור שיר מספריית שירים בשרת, או להפעיל פלייליסט מהשרת (playlist), משמע כל השירים מן הספרייה ישמעו אחד לאחר השני. כל זה נעשה בעזרת חיתוך השיר לחלקים קטנים, המרת החלקים מstring ל bytes, העברתם בעזרת socket מהשרת ללקוח, המרתם חזרה לstring והשמעה של הקובץ בצד הלקוח באופן רציף, ללא המתנה להורדה של החלק הבא או המתנה להורדה של כל השיר. דבר זה יעיל וחוסך בזמן המתנה מיותר.

דרישות פונקציונאליות:

- רשימת דרישות המשתמש מהמערכת (הפעולות בהן נדרשת המערכת לתמוך):
- המערכת צריכה לתמוך במס' רב של משתמשים ובמס' רב של הורדות בו זמנית.
- כמו כן לתמוך במקרים של תקלות ברשת, או הפסקת האינטרנט באמצע הורדת שיר והורדת השיר לאחר החיבור מחדש.
- עבודה מול Buffer :
- ❖ Buffer הוא אזור אחזקה זמני לנתונים בזמן שהוא מחכה להעברה למיקום אחר. זה בדרך כלל ממוקם ב-RAM. הרעיון של buffer פותח כדי למנוע עומס נתונים להיכנס לנמל העברה יוצא.
- ❖ ישנם שימושים נפוצים לbuffer המסייעים בשיפור הביצועים הכוללים של המכשיר. כמעט כל הדיסקים הקשיחים משתמשים בbuffer כדי להקל על אחזור נתונים קל. כל סוג של טיפול בזיכרון ושירות אחסון נתונים ישתמש גם בbuffer כלשהו. אפילו המשימות הבסיסיות ביותר של המעבד הן להשתמש בbuffer להפעלה בצורה של רישומים, שבהם נתונים כמו אופרנדים ומפעילים מאוחסנים לפני שהם מעובדים.
- ❖ לדוגמה, כאשר משתמש מוריד קובץ וידאו או שמע, אחוז מסוים מהקובץ שהורדת מונח בbuffer ומשוחק. עם הפעלת הקליפ, המכשיר מוריד ברציפות את הקובץ ומניח אותו בbuffer. מסיבה זו, יש פחות סיכוי שקובץ הווידאו או השמע יתקע כשמתרחש עומס רשת, אלא אם כן כמובן שקצב ההורדה הוא כה איטי עד כי מהירות ההפעלה עוקפת את מהירות ההורדה.
- ❖ דוגמה נוספת, בעת הדפסת מסמך, כאשר הפקודה PRINT מופעלת על ידי המערכת או היישום, נתוני ההדפסה נשלחים לbuffer ואז מועברים למדפסת. משם, המדפסת ניגשת למידע. זה משחרר את המחשב לביצוע משימות אחרות במהלך ביצוע הפקודה PRINT. חסרון אחד למערכת זו הוא שכל הנתונים בbuffer במהלך תקלה בהתקן אבדים.

בעיות צפויות במהלך הפיתוח ופתרונות (תפעוליות, טכנולוגיות, עומס ועוד):

תיאור הבעיות- הללו כפועל יוצא של דרישות המשתמש מהתוכנה.

- המערכת צריכה להתמודד עם תקלות של חיבור לאינטרנט. • המערכת צריכה להיות בטוחה לשימוש כך שבכל מצב של עבודה היא לא תיפול (תסתיים עבודתה).
- קושי בטיפול ב WAV FILE פירוקו והרכבתו בצורה סריאלית:
1 WAV הוא קובץ שמע, שפותח על ידי MICROSOFT ו- IBM לאחסון audio bitstream במחשב.
- 2 File WAV משתמשת בפורמט RIFF לאחסון מידע בחלקים (chunks). זהו הפורמט העיקרי המשמש במערכות Windows Microsoft עבור שמע גולמי ולא דחוס בדרך כלל, ומפני שהשמע גולמי ולא דחוס הוא תופס יותר מקום מאשר קבצי שמע אחרים (עדכניים יותר) במחשב.
- גודלה של דקה ב WAV אמורה לשקול כ 10MB וגודלו המרבי של קובץ זה יכול להגיע עד 4GB
- 3 הקידוד הרגיל של ביטסטרם הוא פורמט (linear pulse-code modulation - LPCM)
- 4 קבצי WAV בנויים מכותרת (header), בלי אותה כותרת הקובץ לא יהיה תקין ולא יוכל לעבוד כראוי. הכותרת בנויה מ 44 בייטים וכל כמה בייטים ביחד מסמלים חלק בקובץ.

פתרונות אפשריים.

- שינוי באלגוריתם כך שהוא יתמוך במס' רב של קבצים.
- הגנה על הפרויקט ע"י פונקציות מיוחדות של catch i try
- התראה של המערכת במידה ויש נפילה, שימוש במקרה קצה.
- הפעלה של המערכת עם מס' רב של משתמשים איתור תקלות בזמן הפעלה ותיקונם.
- ניסיון להורדת קבצים שונים שאינם wav וכתובת קוד הגנה כך שיראה למשתמש מה התקלה ויאפשר לו ניסיון נוסף בהעלאת שירים.
- טיפול בשגיאה של חוסר חיבור למערכת והצגת שגיאה ללקוח.
- עבודה עם tread תוכל מאוד לעזור במהירות הורדות השירים כך שכמות רבה יותר של שירים תוכל לרדת בו זמנית.

פתרון טכנולוגי נבחר:

טופולוגית הפתרון-

- שרת מבוסס HTTP WEB
- media player
- סגמנטציה
- מערכת Multi-Threading הן בשרת והן בלקוח.

טכנולוגיות בשימוש:

Sound Player Class

ספריה ב-C# מספקת ממשק פשוט לטעינה ונגינה של קובץ WAV.

תומכת בטעינת קובץ WAV מנתיב קובץ, URL, זרם המכיל קובץ WAV או משאב משובץ המכיל קובץ WAV

TCP/IP

פרוטוקול בקרת השידור (TCP) הוא אחד הפרוטוקולים העיקריים של חבילת פרוטוקול האינטרנט. מקורו ביישום הרשת הראשוני בו הוא השלים את פרוטוקול האינטרנט (IP). לכן מכונה בדרך כלל החבילה כולה TCP / IP

TCP מספקת משלוח אמין, מסודר ובודק שגיאות של זרם שמיניות (בתים) בין יישומים הפועלים על מארחים המתקשרים דרך רשת IP. יישומי אינטרנט מרכזיים כמו האינטרנט.

IP מספק את הכתובת של השרת ומוודא שהאתר ישלח לכתובת הייעודית שלו.

thread

Threads- הוא נתיב לביצוע בתוך תהליך. תהליך יכול להכיל מספר threads

Thread מכונה גם תהליך קל משקל. הרעיון הוא להשיג הקבלה על ידי חלוקת תהליך למספר threads. לדוגמה, בדפדפן, כרטיסיות מרובות יכולות להיות פתילים שונים.

ההבדל העיקרי בין תהליך (Process) thread הוא ש threads באותו תהליך פועלים במרחב זיכרון משותף, בזמן שתהליכים פועלים במרחבי זיכרון נפרדים.

WAV File

- WAV File (Waveform Audio File Format/WAV) הוא קובץ שמע, שפותח על ידי IBM - MICROSOFT עבור אחסון audio bitstream. במחשב

WAV File משתמשת בפורמט RIFF לאחסון מידע בחלקים (chunks). זהו הפורמט העיקרי המשמש במערכות Microsoft Windows עבור שמע גולמי ולא דחוס בדרך כלל, ומפני שהשמע גולמי ולא דחוס הוא תופס יותר מקום מאשר קבצי שמע אחרים (עדכניים יותר) במחשב. גודלה של דקה WAV אמורה לשקול כ-10 MB, וגודלו המרבי של קובץ זה יכול להגיע עד 4 GB.

buffer

- Buffer הוא אזור אחזקה זמני לנתונים בזמן שהוא מחכה להעברה למיקום אחר. זה בדרך כלל ממוקם ב-RAM. הרעיון של buffer פותח כדי למנוע עומס נתונים להיכנס לנמל העברה יוצא.

ישנם שימושים נפוצים לbuffer המסייעים בשיפור הביצועים הכוללים של המכשיר. כמעט כל הדיסקים הקשיחים משתמשים בbuffer כדי להקל על אחזור נתונים קל. כל סוג של טיפול נתונים יכול להשתמש בbuffer כדי לשפר את זרימת העבודה שלו.

שפות הפיתוח:

- **C#**: שפת תכנות OOP זו נבחרה לתכנות האפליקציה לפלטפורמת ה-Windows (במקרה זה WPF). הבחירה בשפה זו הייתה די הגיונית וטבעית, שפת C# מצוינת לא רק לפיתוח מורכב, אלא היא גם כוללת בתוכה אינסוף חבילות וכלים מצוינים לפתרון של כמעט כל בעיה. למעשה, בעזרתה אפשר ליצור כמעט כל אפליקציה מכל סוג שהוא. שפת C# הינה גם שפת OOP (מונחת עצמים) מה שמאפשר לנו ליצור תוכנות ואפליקציות חכמות ויעילות. WPF גם מאפשר לנו להשתמש ב-XAML וליצור ממשקי משתמש מתקדמים וגמישים.
- **SQL**: שפת בסיסי נתונים זו נבחרה בגלל הפופולריות שלה והסינטקס היחסית קליל שלה. SQL נתמכת ברוב שירותי הענן הגדולים (AWS, Azure, Google Cloud וכו') מה שהופך את SQL לבחירה האידיאלית לשפת ניהול בסיס הנתונים של הפרויקט.

תיאור הארכיטקטורה הנבחרת:

בפרויקט זה אעבוד עם בארכיטקטורה של שרת לקוח, אשר מיושם ע"י רשימה של משתמשים שמזדהים בכניסה למערכת, המערכת מחוברת לנגן שנבנה בעזרת טכנולוגיית WPF שמאפשר ללקוח לבחור שירים מהשרת לנגן אותם ואף לבנות לעצמו רשימת השמעה. רשימת השירים נלקחת מ-API אשר מחובר לשרת, כל שיר שיוורד במערכת מפוצל לביטים ויוורד בו זמנית אל המערכת (בעזרת thread) כך שנוצרת הורדה מאוד מהירה של אותו שיר. השיר מורכב מחדש ומושמע במהירות. בזמן ניגון השיר המערכת מורידה שירים נוספים אל זיכרון ה-RAM כך שגם במצב של חוסר חיבור עדין המערכת תוכל לפעול ביעילות.

חלוקה לתכניות ומודולים:

הפרויקט מתחלק לכמה חלקים/מודולים:

1. תכנות שרת ובסיסי נתונים - Back End - בחלק זה נתכנת את הלוגיקה של המערכת אשר מתחברת לשרת בשיטת socket
2. תכנות האפליקציות של הפרויקט - Front End - בחלק זה נתכנת את אפליקציית הנגן בצד הלקוח תחת הקפדה על חווית משתמש ועיצוב אחיד.

סביבת השרת (מקומי, וירטואלי, ענן, שירות אירוח)

בפרויקט זה נשתמש בשרת מקומי שנבנה על המחשב כך שהמחשב משתמש גם כשרת וגם כלקוח.

ממשק המשתמש/לקוח - GUI:

ממשקי משתמש/לקוח בפרויקט זה יהיו בנויים ממספר Frameworks ו־XML XAML. הם ממשקי המשתמש/לקוח יהיו אפליקציות לפלטפורמות Windows (במקרה הזה WPF)

ממשקים למערכות אחרות:

Microsoft sql server database בפרויקט זה אנו נשתמש בממשק

שימוש במבני נתונים וארגון קבצים

מבני הנתונים:

מבני הנתונים של פרויקט זה יכללו את המידע הבא:

המידע הכללי של השיר:

מידע כללי על השיר בו הפרטים המזהים שלו, שם השיר ותמונה.

רשימת שירים שהלקוח הוריד:

המערכת שומרת את רשימת השירים שהוריד הלקוח, כמו כן היא מאפשרת ללקוח להוסיף עוד שירים ולהוריד אותם לשרת כך שיכולו להתחבר למערכת כל פעם שיחפצו בכך.

פירוט שיטת האחסון (מאגר, קבצים ובאיזה טכנולוגיה):

רוב הנתונים יאוחסנו בטבלאות בעזרת שפת בסיסי נתונים Sql במערכת Microsoft sql server

חלק מהנתונים יאוחסנו בקבצים נפרדים. הקבצים עצמם ישמרו על השרת בתיקה נפרדת.

מנגנוני התאוששות מפילה/קריסה/תמיכה בטראנזקציות:

מכיוון שמדובר בבסיס נתונים היושב על המחשב ניתן לשער שהסבירות לנפילה של השרת או מחיקה של בסיסי הנתונים, יכולה לקרות כתוצאה מקריסה מוחלטת של המחשב או לחילופין שגיאה בחיבור לשרת (כמו שינוי בכתובת ה IP)

לכן לפני שניגש לנתונים שלנו נבצע מספר בדיקות:

1 החיבור ל database יעשה בעזרת תבנית העיצובית, בצורה כזו נוכל להיות בטוחים שרק חיבור אחד ייווצר.

```
public static class DB
{
    private static SqlConnection _instance;
    public static SqlConnection Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = new SqlConnection(@"Data .....");
            }
            return _instance;
        }
    }
}
```

2. בכדאי להגן על מקרים בהם לא נרשם דבר בתור יוזר נבנה יוזר שהוא נל

וכל פעם שהוא יופיע תופיע הערה ללקוח והחלון ייסגר כך שהלקוח יצטרך לנסות שוב:

```
public static readonly Contact InvalidContact = new Contact(-1, "", "", "");
```

בשורה הזו הגדרתי יוזר שהוא נל.

```
public static bool IsValid(Contact c)
{
    return c != InvalidContact;
}
```

בשורה הזו רשמתי פונקציה שבודקת האם מדובר בשורה שהוכנסו אליה נתונים במידה וכן מחזיר true, כך ניתן לבדוק כל פעם שאנו מריצים פקודת sql האם המשתמש הזין ערכים בתוך השורה.

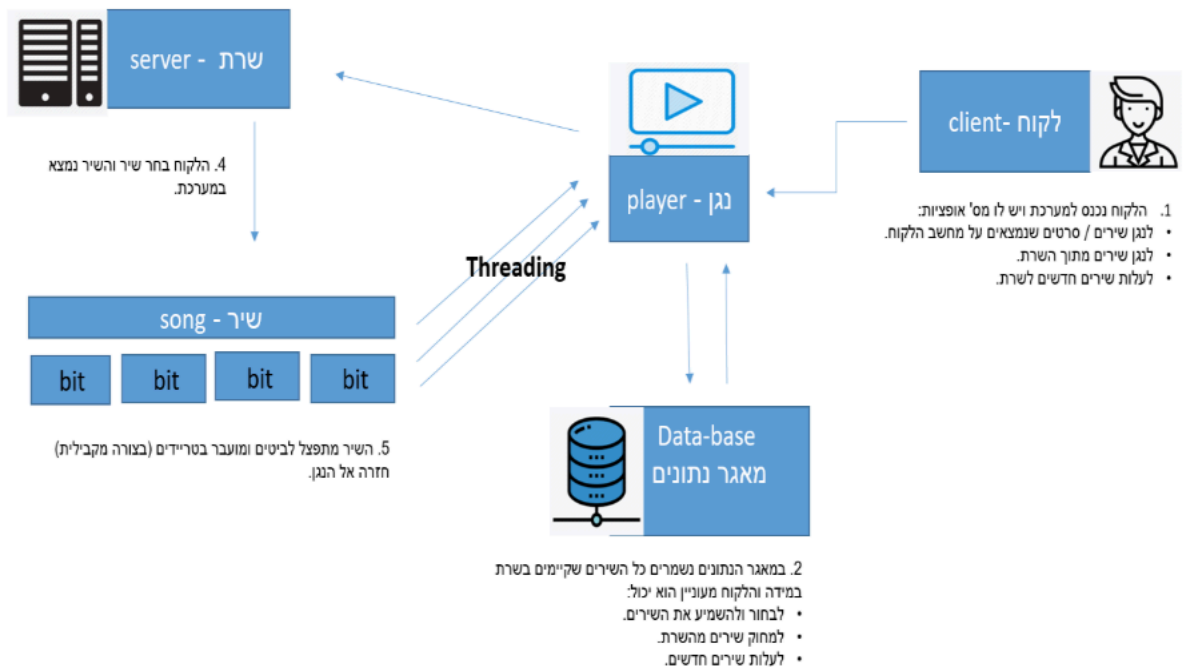
3 כמו כן לאורך כל הקוד השתמשתי בפונקציות של try ו catch כך שבמידה ויש נפילות ישנה התרעה ללקוח מדוע נפלה המערכת.

תרשימי מערכת מרכזיים

Use Case

Sequence diagram - רצף קריאות פונקציות מרכזיות בלוגיקה העסקית
המרכזית של הפרויקט :

Data flow





הפונקציות בנגן

StartBtn - לחיצה על הפעלת הנגן	Open_folder_Click - פתיחת תיקיה חדשה
StopBtn - עצירת הנגן	me_media_ope - אתחול של הקובץ
PosSlider_MouseDown - השעיה של הנגן	Open_file_Click - פתיחת קובץ יחיד (להבדיל מתיקה)
speed_ValueChanged - הגברת מהירות הנגן.	Playtrack - מנגן את הקובץ
volume_slider_ValueChanged - שינוי בווליום (הגברה והנמכה).	media_end_event - מריץ שיר חדש לאחר ששיר אחד נגמר (בלולאה).
CloseApp_click - סגירת הנגן	PosSlider_MouseUp - עצירה באמצע שיר.
speed_ValueChanged - מהירות הנגן	Time_tick - פונקציה שמטרתה לחלק את הביטים למקטעים ובכך למדוד איפה נעצר באמצע המקטע.
PlayPlaylist - פונקציה שגורמת לנגן להמשיך לנגן לטרק השני.	
postion_vol_ValueChanged	
balance_vol_ValueChanged - כיון רמקולים לכיוונים (רמקול ימין ושמאל)	
Media_MediaFaield - הודעת שגיאה ברגע שאין קובץ	

תיאור המרכיב האלגוריתמי - חישובי

במסגרת הפרויקט נשתמש במספר רב של אלגוריתמים ברמות השונות של הפרויקט:

- **LPCM (linear pulse-modulated)** הוא פורמט bitstream הקידוד הרגיל של קובצי .WAV.
- **אלגוריתם לקידוד קובצי WAV** בנויים מכותרת (header), בלי אותה כותרת הקובץ לא יהיה תקין ולא יוכל לעבוד כראוי. הכותרת בנויה מ-44 ביטים (bytes) וכל כמה ביטים ביחד מסמלים חלק בקובץ, והחלקים הם:
 - **בייטים 1-4:** מסמן את הקובץ כקובץ riff, אורכו של כל תו הוא בייט אחד.
 - **בייטים 5-8:** גודל הקובץ הכולל - 8 בתים, בתים (מספר שלם של 32 סיביות). בדרך כלל, תמלא זאת לאחר היצירה.
 - **בייטים 9-12:** כותרת סוג קובץ. לענייננו זה תמיד שווה ל-"WAVE".
 - **בייטים 13-16:** פורמט chunk marker כולל null נגרר.
 - **בייטים 17-20:** אורך נתוני הפורמט כמפורט מעלה.
- **Player Information Logger** - הקוד ממיר אותו למספר.

אלגוריתם סגמנטציה -

מטרת האלגוריתם:

- חלוקת שיר מקורי למקטעים קצרים ("סגמנטים") באורך 2 שניות כל אחד.
- העברת הסגמנטים מהשרת ללקוח דרך Socket.
- סגירת ה-Socket באופן מבוקר על מנת להבטיח רצף שירים חלק ללא המתנה מיותרת.
- אפשרות עצירת השמעת השיר הנוכחי באמצע הסגמנט.
- שחזור קובץ WAV תקין מכל סגמנט.

תיאור מפורט של האלגוריתם:

1. **חיתוך השיר המקורי לסגמנטים:**
 - השיר המקורי מחולק למקטעים באורך 2 שניות כל אחד.
 - כל סגמנט מקבל מספור סידורי.
2. **שליחת סגמנטים דרך Socket:**
 - כל סגמנט נשלח מהשרת ללקוח דרך Socket פתוח.
 - יחד עם כל סגמנט נשלחת גם כותרת המכילה מידע על הסגמנט, כגון:
 - מספור סידורי
 - אורך הסגמנט
 - נתוני שמע
 - הכותרת חשובה לשחזור קובץ WAV תקין מהסגמנטים.
3. **סגירת Socket באופן מבוקר:**
 - ה-Socket ייסגר בתנאים מסוימים, כגון:
 - קבלת הוראה מהלקוח לעצור את השיר.
 - השלמת העברת כל הסגמנטים של השיר.
 - סגירת ה-Socket תגרום להפסקת העברת הנתונים מהשרת ללקוח.
 - כתוצאה מכך, נגן המדיה בלקוח יפסיק לנגן את השיר הנוכחי.
4. **עצירת השמעת השיר באמצע הסגמנט:**
 - ניתן לעצור את השמעת השיר הנוכחי בכל עת על ידי שליחת הוראה מתאימה מהלקוח לשרת.
 - קבלת ההוראה תגרום לשרת לסגור את ה-Socket באופן מיידי.
 - כתוצאה מכך, נגן המדיה בלקוח יפסיק לנגן את השיר הנוכחי באמצע הסגמנט הנוכחי.
5. **שחזור קובץ WAV מהסגמנטים:**
 - ניתן לשחזר קובץ WAV תקין מהסגמנטים על ידי:
 - מיון הסגמנטים לפי מספרם הסידורי.
 - חיבור נתוני השמע של כל סגמנט ברצף.
 - הוספת כותרת WAV מתאימה.
 - קובץ ה-WAV המוחזר יכיל את השיר המקורי באיכות זהה לשיר המקורי.

יתרונות האלגוריתם:

- **הזרמת שירים חלקה:** השיר הבא מתחיל לנגן מיד לאחר סיום השיר הקודם, ללא המתנה מיותרת.
- **חיסכון ברוחב פס:** ניתן להפסיק את העברת השיר הנוכחי בכל עת, ובכך לחסוך ברוחב פס.
- **גמישות:** ניתן לחתוך שירים למקטעים קצרים יותר או פחות לפי הצורך.
- **שחזור קל:** ניתן לשחזר בקלות קובץ WAV תקין מהסגמנטים.

חסרונות האלגוריתם:

- **מורכבות:** האלגוריתם מורכב יותר מאשר נגינת שיר מקובץ WAV רגיל.
- **עומס על השרת:** השרת צריך לשלוח את הסגמנטים ללקוח בזמן אמת, מה שעלול להגביר את העומס עליו.
- **תלות ב-Socket:** האלגוריתם תלוי ב-Socket פתוח ותקין לצורך העברת הסגמנטים.

תיאור/התייחסות לנושאי אבטחת מידע

פרוטוקול העברת המידע הנבחר בין הנגן לבין השרת הוא מסוג HTTP שנותן שכבת הגנה נוספת להעברת המידע בין החומרה לענן ולהפך.

- בכל אחת מהאפליקציות של השירות יש מנגנון אבטחה שבודק האם המשתמש רשום לשירות והאם הוא רשאי ולהשתמש בשירות.

הבדיקה נעשית על ידי שליחת שאילתה לבסיס נתונים של משתמשים בשירות ובדיקת התאמה של שם המשתמש והסיסמה.

פתרונות לנושאי אבטחת מידע

- פרוטוקול העברת המידע הנבחר בין הנגן לבין השרת הוא מסוג HTTP שנותן שכבת הגנה נוספת להעברת המידע בין החומרה לענן ולהפך.

- בכל אחת מהאפליקציות של השירות יש מנגנון אבטחה שבודק האם המשתמש רשום לשירות והאם הוא רשאי ולהשתמש בשירות.

הבדיקה נעשית על ידי שליחת שאילתה לבסיס נתונים של משתמשים בשירות ובדיקת התאמה של שם המשתמש והסיסמה.

משאבים הנדרשים לפרויקט:

מספר שעות המוקדש לפרויקט : 7 שעות בשבוע.

ציוד נדרש: 2 מחשבים מבוססי Windows

תוכנות נדרשות:

- Visual Studio Code, Visual Studio 2019: תוכנות לכתיבה ועריכת קוד
- DBaiver: כלי עריכת בסיסי נתונים
- Microsoft SQL Server: מערכת ניהול בסיסי נתונים
- Chrome Browser: דפדפן אינטרנט

ידע חדש שנדרש ללמוד לצורך ביצוע הפרויקט:

- קורס מתקדם בשפת C#
- קורס מתקדם בשפת WPF
- ידע בבסיסי נתונים
- ידע בתקשורת נתונים
- יכולות אלגוריתמיות גבוהות

ספרות ומקורות מידע:

- קורסים ב-YouTube ובאתר Lynda
- לימודים פרונטליים בכיתה
- כתבות באינטרנט בנושא שרתים ופרוטוקולים
- קורסים ב-Udemy

תכנון הבדיקות שיבוצעו:

בדיקות תהליכיות ברמת משתמש בהן נדרשת המערכת לעמוד (full Flow)

בדיקות יחידה למודולים המרכזיים בהן נדרשת המערכת לעמוד. (test unit)

הפעלת המערכת	המערכת צריכה להתחבר באופן מסודר לשרת
הצגת רשימת שירים	לאחר ההתחברות היא צריכה להריץ שירים באופן רציף
העלאת כמה שירים במקביל והפסקת האינטרנט	נבדוק שניתן להוריד כמה שירים במקביל מהמערכת ובזמן ההורדה השירים הנוספים יורדים.
הוספה / מחיקה של שיר	המערכת מוסיפה שירים לשרת וכל התחברות נותנת אופציה ללקוח לנגן את אותם שירים כמו כן להיפך היא נותנת ללקוח אופציה למחוק שירים.
הוספת אירוע המערכת התנתקה	במצב של ניתוק המערכת מהשרת יופיע למשתמש הודעה כי הוא מנותק מהשרת וכי כאשר יחובר מחדש ההורדה תחל שוב
התחברות לשירות	בכניסה לכל אחת מהאפליקציות של השירות, תבוצע בדיקה של שם המשתמש והסיסמה של המשתמש המנסה להיכנס לאפליקציה. הבדיקה תבצע מול בסיס נתונים של משתמשים שרשומים לשירות.

הגדרת הבעיה:

המערכת תנהל את דרך ההורדה של השירים ותאפשר הורדה של מספר קבצי אודיו בו-זמנית ע"י שימוש בסגמנטציה (חלוקה דיגיטלית למקטעים) של קבצי ה.WAV.

דבר זה יכול למנוע מצבים של קטיעה בשירים ואף לאפשר האזנה או צפייה לקובץ ללא תלות בחיבור אינטרנטי. אזי המערכת תצטרך להתמודד עם הורדה של מידע באופן חד ערבי (כאשר כל פעם יורד חלק אחד מהמידע / שיר) להעברת מידע בצורה מקבילית (ביטים אשר יורדים במקביל).

בנוסף המערכת תתעד את היסטוריית השינויים אשר קרו במערכת, כלומר יהיה ניתן לתחקר שינויים אשר קרו בשוגג ולהחזיר את המערכת למצב שהיה בעבר. כמו כן לפי שינויים אלו המערכת תוכל לבדוק מה השתנה וגרם לשינויים החריגים הללו, לדוגמא: תקלת חיבור למערכת או העלאה לא תקינה של קובץ השיר, ובעזרת תיעודיים אלו המערכת תוכל להראות למשתמש מה התקלה ובכך תינתן לו אפשרות נוספת להעלאת השירים/ להתחבר אל המערכת.

את הבעיות האלו המערכת תפתור בעזרת האלגוריתמים הבאים:

אלגוריתם לקידוד WAV file - הקידוד הרגיל של bitstream הוא פורמט linear pulse-code modulation (LPCM).

- קובצי WAV בנויים מכותרת (header), (כותרת הקובץ לא יהיה תקין ולא יוכל לעבוד כראוי. הכותרת בנויה מ-44 ביטים (bytes) כמה ביטים ביחד מסמלים חלק בקובץ, והחלקים הם: ביטים 1-4: מסמן את הקובץ כקובץ riff, ואורכו של כל תו הוא בייט אחד.
- ביטים 5-8: גודל הקובץ הכולל - 8 בתים, בתים (מספר שלם של 32 סיביות). בדרך כלל, תמלא זאת לאחר היצירה.
- ביטים 9-12: כותרת סוג קובץ. לענייננו זה תמיד שווה ל "WAVE".
- ביטים 13-16: פורמט chunk marker כולל null נגרר.
- ביטים 17-20: אורך נתוני הפורמט כמפורט מעלה
- הקוד ממיר אותו למספר.

אלגוריתם סגמנטציה -

- כל סגמנט יהיה בקפיצה של 2 שניות מהשיר המקורי.
- ה socket שעליו הסגמנטים עוברים מהשרת ללקוח ייסגר בתנאי מסוים, בכדי לגרום לשירים לפעול אחד לאחר השני ללא המתנה מיותרת. כך גם אפשר לגרום לבאפר להפסיק את ההעברה באמצע במידה ולא רוצים להמשיך לשמוע את השיר המדובר.
- נרכיב מחדש את ה header עבור כל סגמנט לקובץ (WAV) יש כותרת המגדירה אותו ().

תהליכים עיקריים בפרויקט:

לצורך הקמת המערכת עבור הפיתוח אנו נשתמש במחשב אשר ישמש גם כ שרת המקומי וגם כלקוח ן במסד נתונים. מנהל המערכת יהיה המשתמש החזק ביותר הוא היחיד ש שתינתן לו אפשרות לבצע פעולות על המערכת(כלומר שינויים במערכת), כל שינוי אשר יתקיים במערכת ע"י משתמש רגיל יצטרך לקבל את אישורו של המנהל. במסד הנתונים ישמרו כל הנתונים אודות המשתמשים (לדוגמא: רשימת השירים ש הלקוח הוריד, שם המשתמש והסיסמא) ובנוסף המסד ישמור מידע כללי על השירים בו הפרטים המזהים שלו, שם השיר ותמונה , מסד הנתונים ישמור גם את רשימת ההשמעה של המשתמש. ובנוסף לכך מסד הנתונים ישמור תיעודיים אודות השינויים אשר בוצעו במערכת לצורכי בקרה. כמו כן המידע הרגיש אודות המשתמשים יועבר מוצפן, לצורך אבטחת המידע.

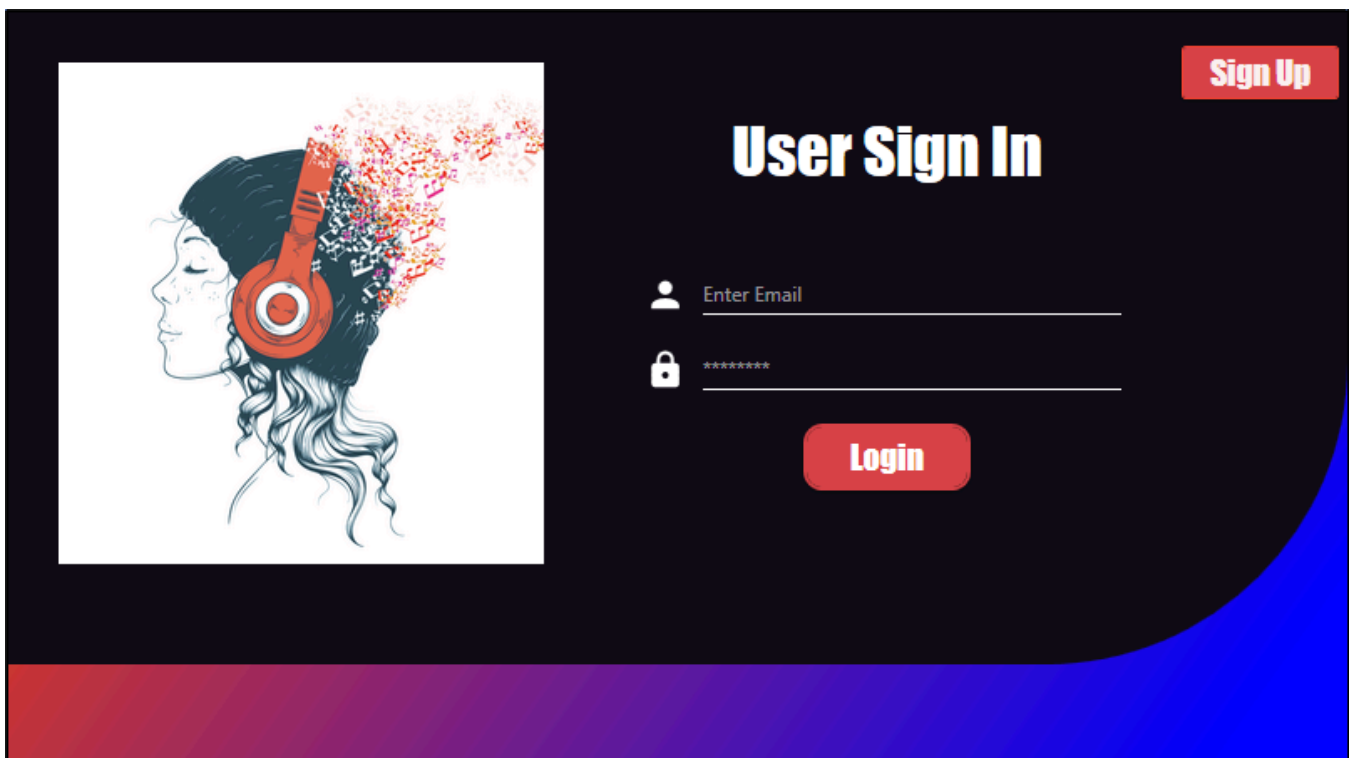
הפעלת המערכת:

מסך זה הוא מסך אשר אינו מופיע למשתמש אך פועל ברקע מסך זה הוא המסך של השרת אשר מחכה להודעה מהלקוח הודעה היא בעצם אישור חיבור(הרשמה / רישום ובחירת שיר מן הגלריה).

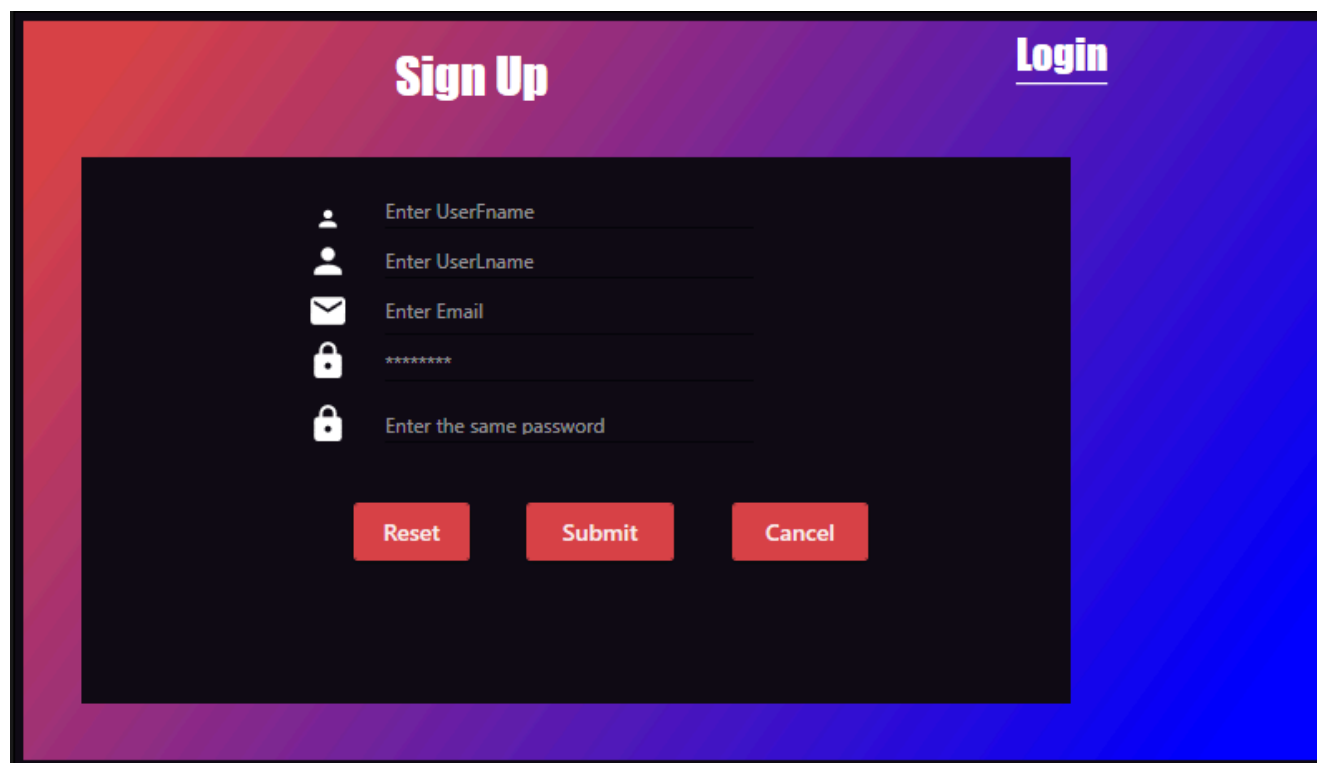
```
D:\users\liam arbelani\שולחן עבודה\הצעת פרויקט של גל\FilesServer\FilesServer\bin\Debug\FilesServer.exe
the server is running at port:127.0.0.1:8001
waiting for a connection
connection accepted
the message from client side

The directory was created successfully at 31/12/2023 15:38:46.
335296
end of segment 1 from 168
end of segment 2 from 168
end of segment 3 from 168
end of segment 4 from 168
end of segment 5 from 168
end of segment 6 from 168
end of segment 7 from 168
end of segment 8 from 168
end of segment 9 from 168
end of segment 10 from 168
end of segment 11 from 168
```

להלן המסך הראשון אשר מופיע עבור כניסה למערכת של משתמש קיים, המשתמש מזין את כתובת המייל והסיסמא איתה הוא נרשם למערכת



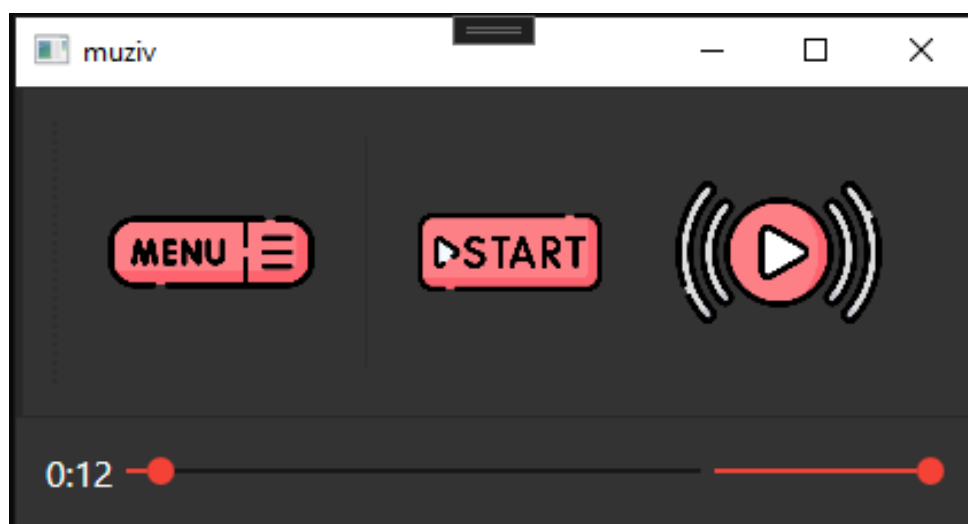
להלן מסך ההרשמה למערכת של משתמש חדש במסך זה המשתמש מזין את הפרטים הנדרשים על מנת להכנס למערכת.



The image shows a 'Sign Up' form with a dark background and a gradient border. The form contains the following elements:

- Sign Up** (Title)
- Login** (Link)
- Enter UserName (with user icon)
- Enter UserLname (with user icon)
- Enter Email (with envelope icon)
- ***** (password field with lock icon)
- Enter the same password (with lock icon)
- Reset** button
- Submit** button
- Cancel** button

להלן המסך של ממשק המערכת מסך זה הוא המסך אשר ממנו בלחיצה על כפתור התיקיה תיפתח הגלריה.



להלן המסך השלישי מסך הגלריה אשר ממנו בוחרים שיר וממשק המערכת של הנגן לאחר הבחירה
אנו ננגן אותו.



קוד הפרויקט:

צד הלקוח – מבוסס WPF:

```
1 <Application x:Class="MediaPlayerWPF.App"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:local="clr-namespace:MediaPlayerWPF"
5     StartupUri="SignInWindow.xaml">
6 <Application.Resources>
7 <ResourceDictionary>
8 <ResourceDictionary.MergedDictionaries>
9 <ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component
10 <ResourceDictionary Source="pack://application:,,,/MaterialDesignThemes.Wpf;component
11 <ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/The
12 <ResourceDictionary Source="pack://application:,,,/MaterialDesignColors;component/The
13 </ResourceDictionary.MergedDictionaries>
14 </ResourceDictionary>
15 </Application.Resources>
16 </Application>
```

פותח את מסך כניסת המשתמש ישר עם הרצת האפליקציה

main window code:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Windows;
using System.Windows.Controls.Primitives;
using System.Windows.Input;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Diagnostics;
using System.ComponentModel;

namespace MediaPlayerWPF
{
    /// <summary>
    /// Interaction Logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        private int counter=0; //play segments סופר את
        private ASCIIEncoding asciiEnc = new ASCIIEncoding();
    }
}
```

```

// strings to byte arrays קוד להמיר ASCII

// לסכימת הסגמנטים של קבצי השמע
private Queue<List<byte>> segmentsQueue = new Queue<List<byte>>();

// דגל שקובע אם הניגון הפסיק
private bool finish = false;

// דגל שקובע אם הנגן כעת מנגן משהו
private bool mediaPlayerIsPlaying = false;

// הסוקט ליצירת הקשר עם הסרבר
private Socket socket = null;

// IP address and endpoint לסרבר
private IPAddress hostAddress = null;
IPEndPoint hostEndPoint = null;

// Reference to the secondary window
SecondWindow window;

// דגל שקובע אם השיר הופסק באמצע
private bool IsPause = true;

private ParameterizedThreadStart downloadThreadStart = null;
private Thread downloadThread = null;
private ThreadStart playThreadStart = null;
private Thread playThread = null;
private bool Isplaying = false;
//private bool CanContinue = true;
private int next = 0;
//private TcpListener listener = null;
public MainWindow()
{
    InitializeComponent();

    //DispatcherTimer timer = new DispatcherTimer();
    //timer.Interval = TimeSpan.FromSeconds(1);
    //timer.Tick += timer_Tick;
    //timer.Start();

    try
    {
        // פותח שרת טיסיפי על אייפי ופורט ספציפיים
        hostAddress = IPAddress.Parse("127.0.0.1");
    }
}

```

```

        hostEndPoint = new IPEndPoint(hostAddress, 8001);
        socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
        socket.Connect(hostEndPoint);

        // יוצר אובייקט חלון שני
        window = new SecondWindow(socket);

        //Console.WriteLine("Conneting...");

    }
    catch (Exception e)
    {
        socket.Close();
        //Console.WriteLine("Error..." + e.StackTrace);
    }
}

private void OnSecondWindowButtonClicked(object sender, RoutedEventArgs e)
{
    // איבנט שמעביר לחלון השני במידה ויש לחיצה על הכפתור
    if (window == null || window.IsClosed)
    {
        window = new SecondWindow(socket);
    }
    window.WindowState = WindowState.Maximized;
    window.Show();
}

private void timer_Tick(object sender, EventArgs e)
{
    //if ((mePlayer.Source != null) && (mePlayer.NaturalDuration.HasTimeSpan) && (!userIsDraggingSlider))
    //{
    //     sliProgress.Minimum = 0;
    //     sliProgress.Maximum = mePlayer.NaturalDuration.TimeSpan.TotalSeconds;
    //     sliProgress.Value = mePlayer.Position.TotalSeconds;
    //}
}

private void Open_CanExecute(object sender, CanExecuteRoutedEventArgs e)
{
    //e.CanExecute = true;
}

private void Open_Executed(object sender, ExecutedRoutedEventArgs e)
{

```



```

        //OpenFileDialog openFileDialog = new OpenFileDialog();
        //openFileDialog.Filter = "Media files
(*.mp3;*.mpg;*.mpeg)|*.mp3;*.mpg;*.mpeg|All files (*.*)|*.*";
        //if (openFileDialog.ShowDialog() == true)
        //{
            mediaPlayer.Source = new Uri(openFileDialog.FileName);

        //}
    }
    protected override void OnClosed(EventArgs e)
    {
        base.OnClosed(e);
        Environment.Exit(0);
        //Application.Current.Shutdown();
    }
    private void Play_CanExecute(object sender, CanExecuteRoutedEventArgs e)
    {
        // e.CanExecute = (mediaPlayer != null) && (mediaPlayer.Source != null);
        //
        e.CanExecute = true;
    }

    private void Play_Executed(object sender, ExecutedRoutedEventArgs e)
    {
    }

    private void Pause_CanExecute(object sender, CanExecuteRoutedEventArgs e)
    {
        e.CanExecute = mediaPlayerIsPlaying;
    }

    private void Pause_Executed(object sender, ExecutedRoutedEventArgs e)
    {
        mediaPlayer.Pause();
    }

    private void Stop_CanExecute(object sender, CanExecuteRoutedEventArgs e)
    {
        e.CanExecute = mediaPlayerIsPlaying;
    }

    private void Stop_Executed(object sender, ExecutedRoutedEventArgs e)
    {
        mediaPlayer.Stop();
        mediaPlayerIsPlaying = false;
    }

```

```

}

private void sliProgress_DragStarted(object sender, DragStartedEventArgs e)
{
    //userIsDraggingSlider = true;
}

private void sliProgress_DragCompleted(object sender, DragCompletedEventArgs e)
{
    //userIsDraggingSlider = false;
    //mePlayer.Position = TimeSpan.FromSeconds(sliProgress.Value);
}

private void sliProgress_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
{
    //    LblProgressStatus.Text =
    TimeSpan.FromSeconds(sliProgress.Value).ToString(@"hh\:mm\:ss");
}

private void Grid_MouseWheel(object sender, MouseWheelEventArgs e)
{
    //mePlayer.Volume += (e.Delta > 0) ? 0.1 : -0.1;
}

private void DownloadAsync(Object obj)
{
    // פותח סוקט
    using (Socket socket = obj as Socket)
    {
        object sender = new object();
        RoutedEventArgs e = new RoutedEventArgs();
        byte[] binDataOut;

        string str = window.GetSongName(sender, e);
        binDataOut = asciiEnc.GetBytes("1$" + str);
        // שולח את שם השיר לשרת
        socket.Send(binDataOut, 0, binDataOut.Length, SocketFlags.None);
        byte[] binDataIn = new byte[255];
        // מקבל מהשרת את אורך השיר
        int k = socket.Receive(binDataIn);
        int songLength = Int32.Parse(asciiEnc.GetString(binDataIn, 0, k));

        binDataIn = new byte[255];
    }
}

```

```

List<byte> lsbytes = new List<byte>();
Thread.Sleep(150); // עוצר את התוכנית ל 150 מיליסקנד

k = socket.Receive(binDataIn, 0, 255, SocketFlags.None);
// מקבל מהסרבר את גודל הקובץ אותו מורידים
string info = asciiEnc.GetString(binDataIn, 0, k);
int segments = int.Parse(info.Split('#')[1]);
bool stop = false;
// עוברים על גודל הסגמנטים
for (int i = 0; i < segments && !stop; i++)
{
    if (next > 1)
    {
        next = 1;
        finish = true;
        //CanContinue = false;
        //return;
    }
    // מקבלים סוג סגמנט
    k = socket.Receive(binDataIn, 0, 255, SocketFlags.None);
    while (k >= 255)
    {
        // מקבלים את הסגמנט עצמו
        lsbytes.AddRange(binDataIn);
        k = socket.Receive(binDataIn, 0, 255, SocketFlags.None);
    }
    while (k != 3)
    {
        // מקבלים את הסגמנט עצמו
        lsbytes.AddRange(binDataIn);
        k = socket.Receive(binDataIn, 0, 255, SocketFlags.None);
    }

    // משרשים את הסגמנט לתור סגמנטים
    segmentsQueue.Enqueue(new List<byte>(lsbytes));
    //Console.WriteLine(lsbytes.ToArray().Length);
    lsbytes.Clear();

    string status= "end of segment " + (i + 1) + " from " + segments;
    byte [] binData = asciiEnc.GetBytes(status);
    // שולח שקיבל הכל
    socket.Send(binData);

    //Console.WriteLine("end of segment " + (i + 1) + " from " + segments);
}
}

```

```

        finish = false;
    }

    private void Play()
    {
        counter = 0;
        while (!finish)
        {
            // עובר על הסגמנטים
            while (segmentsQueue.Count > 0)
            {
                // אם עצר לא עושה כלום
                if (IsPause)
                {
                    // מפעיל את הסגמנט
                    List<byte> lsbytes = segmentsQueue.Dequeue();
                    using (var ms = new MemoryStream(lsbytes.ToArray()))
                    {
                        System.Media.SoundPlayer soundPlayer = new
System.Media.SoundPlayer(ms);
                        if (!finish) soundPlayer.PlaySync();
                        //can also use soundPlayer.PlaySync()
                    }
                    counter++;
                    Debug.WriteLine("end playing of segment "+ counter);
                }
            }
            finish = false;
            counter = 0;
        }
    }

    //הופך את המשתנה ל פולס כדי לעצור את השמע
    private void DoPause(object sender, RoutedEventArgs e)
    {
        IsPause = !IsPause;
    }
}

```

music Gallery (secondWindow) code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace MediaPlayerWPF
{
    /// <summary>
    /// Interaction Logic for SecondWindow.xaml
    /// </summary>
    public partial class SecondWindow : Window
    {
        private Socket socket = null;
        private ASCIIEncoding asciiEnc = new ASCIIEncoding();
        string songname = "empty";
        public SecondWindow()
        {
            InitializeComponent();
        }

        public SecondWindow(Socket socket)
        {
            InitializeComponent();
            this.socket = socket;
            DownloadGallery();
        }

        private void DownloadGallery()
        {
            שולח לסרבר לקבל שמות של שירים
            byte[] binDataOut;
            binDataOut = asciiEnc.GetBytes("2$names");
            socket.Send(binDataOut, 0, binDataOut.Length, SocketFlags.None);
        }
    }
}
```

```

byte[] binDataIn = new byte[255];

//מקבל את שמות השירים
int k = socket.Receive(binDataIn);
string [] songNames = asciiEnc.GetString(binDataIn, 0, k).Split('$');
songNames = songNames.Select(x => x.Replace("jpg", "wav")).ToArray();
Console.WriteLine(songNames);

List <ImageSource > lsImgs= DownloadAllImgs();
CreateGridGrallery(songNames, lsImgs);
}

private void CreateGridGrallery(string[] songNames, List<ImageSource> lsImgs)
{
    //יוצר גריד של שירים
    Grid grid = this.mygrid;
    grid.ColumnDefinitions.Add(new ColumnDefinition());
    grid.ColumnDefinitions.Add(new ColumnDefinition());
    int row = 0;
    for (int i = 0; i < songNames.Length; i+=2)
    {
        //מוסיף שורה לשורות הגרידים
        grid.RowDefinitions.Add(new RowDefinition());
        int col = 0;
        for (int j = i; j < i+2; j++)
        {
            //מוסיף טור לגרידים
            grid.ColumnDefinitions.Add(new ColumnDefinition());

            //יוצר תמונה וסטאק פאנל לשם תמונה
            StackPanel sp = new StackPanel();
            //הגדרות של תמונה וכפתור
            Image im = new Image();
            im.Width = 100;
            im.Height = 100;
            im.Source = lsImgs[j];
            Button b = new Button();
            b.Content = im;
            b.ToolTip = songNames[j];
            b.Width = 150;
            b.Height = 150;
            b.Click += B_Click;
            sp.Children.Add(b);
            //מוסיף לייבל עם השם של השיר
            Label l = new Label();
            l.Content = songNames[j];

```

```

        sp.Children.Add(1);
        sp.Margin = new Thickness(10);
        Grid.SetRow(sp, row);
        Grid.SetColumn(sp, col++);
        // מוסיף לגריד ולשורה וטור הנכונים
        mygrid.Children.Add(sp);

    }
    row++;

}

}

private void B_Click(object sender, RoutedEventArgs e)
{
    // לחיצה על כפתור משנה את שם השיר הנוכחי וסוגרת את החלון
    songname = (sender as Button).ToolTip.ToString();
    Close();
}

// מוריד את כל התמונות של השירים
private List<ImageSource> DownloadAllImgs()
{
    List<ImageSource> imgs = new List<ImageSource>();
    ImageSource src = null;
    byte[] binDataOut;
    do
    {
        byte[] binDataIn = new byte[255];
        binDataOut = asciiEnc.GetBytes("getnext");
        socket.Send(binDataOut, 0, binDataOut.Length, SocketFlags.None);
        src = DownloadsingleImg();
        if (src != null)
            imgs.Add(src);

    } while (src != null);

    binDataOut = asciiEnc.GetBytes("done");
    // שולח לסרבר שקיבל את כל התמונות
    socket.Send(binDataOut, 0, binDataOut.Length, SocketFlags.None);
    return imgs;
}

```

```

}

private ImageSource DownloadsingleImg()
{
    try
    {
        List<byte> lsbytes = new List<byte>();
        Thread.Sleep(100);
        byte[] bufferIn = new byte[255];
        //מקבל מהסרבר את השירים
        int k = socket.Receive(bufferIn, 0, 255, SocketFlags.None);
        //DONE!!
        if (k == 1)
            return null;
        while (k >= 255)
        {
            lsbytes.AddRange(bufferIn);
            //מקבל מהסרבר את המשך השירים
            k = socket.Receive(bufferIn, 0, 255, SocketFlags.None);
        }

        byte[] temp = new byte[k];
        Array.Copy(bufferIn, temp, k);
        lsbytes.AddRange(temp);
        //myimage.Source = ToImage(lsbytes.ToArray());
        byte[] imgdata = lsbytes.ToArray();
        System.Drawing.Image tempImg = Helper.MyByteArrayToImage(imgdata);
        return Helper.MyToImageSource(tempImg);
    }
    catch (Exception)
    {
        return null;
    }
}

protected override void OnClosed(EventArgs e)
{
    base.OnClosed(e);
    this.IsClosed = true;
}

public string GetSongName(object sender, RoutedEventArgs ee)

```



```

    {
        return songname;
    }
    public bool IsClosed { get; private set; }
}
}

```

sign in window code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Data.OleDb;
using System.Data;
using System.Text.RegularExpressions;
namespace MediaPlayerWPF
{
    /// <summary>
    /// Interaction logic for SignInWindow.xaml
    /// </summary>
    public partial class SignInWindow : Window
    {

        public SignInWindow()
        {
            InitializeComponent();
        }

        private void Border_MouseDown(object sender, MouseButtonEventArgs e)
        {
            //DragMove();
        }
        private void BtnExit_Click(object sender, RoutedEventArgs e)

```

```

{
    // עובר לעמוד סיון אפ
    SignUPWindow signUPWindow = new SignUPWindow();
    signUPWindow.Show();
    this.Close();
}

private void BtnLogin_Click(object sender, RoutedEventArgs e)
{
    // עושה את הלוגין
    if (txtEmail.Text.Length == 0)
    {
        // לייבל של איימיל
        errorMessage.Text = "Enter an email.";
        txtEmail.Focus();
    }
    else if (!Regex.IsMatch(txtEmail.Text,
@"^[a-zA-Z][\w\.-]*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.-]*[a-zA-Z]
$"))
    {
        // בודק אם האיימיל וליד עם רגאקס
        errorMessage.Text = "Enter a valid email.";
        txtEmail.Select(0, txtEmail.Text.Length);
        txtEmail.Focus();
    }
    else
    {
        // מתחבר לדיבי ובודק אם המשתמש קיים
        string email = txtEmail.Text;
        string password = txtPassword.Password;
        OleDbConnection con_stringin = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\project_gal\FileServer\FileServer\FileShareDB.accdb");
        con_stringin.Open();
        OleDbCommand cmd = new OleDbCommand("Select*from users where Email='" +
email + "' and password='" + password + "'", con_stringin);

        cmd.CommandType = CommandType.Text;
        OleDbDataAdapter adapter = new OleDbDataAdapter();
        adapter.SelectCommand = cmd;
        DataSet dataSet = new DataSet();
        adapter.Fill(dataSet);
        if (dataSet.Tables[0].Rows.Count > 0)
        {
            errorMessage.Text = "Success";
            //string username = dataSet.Tables[0].Rows[0]["FirstName"].ToString()

```

```

+ " " + dataSet.Tables[0].Rows[0]["LastName"].ToString();
        // welcome.TextBlockName.Text = username;//Sending value from one form
to another form.

        // welcome.Show();
        // Close();

        // אם משתמש קיים לעבור למיין וינדו
        MainWindow main = new MainWindow();
        main.Show();
        this.Close();
    }
    else
    {
        errorMessage.Text = "Sorry! Please enter existing emailid/password.";
    }
    con_stringin.Close();
}
}
private void buttonRegister_Click(object sender, RoutedEventArgs e)
{
    //עבור לסיינאפ בלחיצה
    SignUPWindow reg = new SignUPWindow();
    reg.Show();
    Close();
}
}
}

```

sign up window code:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Data;

```

```

using System.Data.OleDb;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.IO;
namespace MediaPlayerWPF
{
    /// <summary>
    /// Interaction Logic for SignUPWindow.xaml
    /// </summary>
    public partial class SignUPWindow : Window
    {
        public SignUPWindow()
        {
            InitializeComponent();
        }

        private void SignIn_Click(object sender, RoutedEventArgs e)
        {
            // לעבור לסיין אין בלחצה
            SignInWindow login = new SignInWindow();
            login.Show();
            Close();
        }

        private void button2_Click(object sender, RoutedEventArgs e)
        {
            // בלחיצה עושה ריסט לפילדים
            Reset();
        }
        public void Reset()
        {
            // עושה ריסט לפילדים

            textBoxFirstName.Text = "";
            textBoxLastName.Text = "";
            textBoxEmail.Text = "";

            passwordBox.Password = "";
            passwordBoxConfirm.Password = "";
        }

        private void button3_Click(object sender, RoutedEventArgs e)
        {
            // סוגר חלון
            Close();
        }
    }
}

```

```

private void Submit_Click(object sender, RoutedEventArgs e)
{
    // מפעיל סיין אפ
    if (textBoxEmail.Text.Length == 0)
    {
        // בודק אם כתב מייל
        errorMessage.Text = "Enter an email.";
        textBoxEmail.Focus();
    }
    else if (!Regex.IsMatch(textBoxEmail.Text,
@"^[a-zA-Z][\w\.-]*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.-]*[a-zA-Z]$"
))
    {
        // בודק אם המייל תקין עם רגאקס
        errorMessage.Text = "Enter a valid email.";
        textBoxEmail.Select(0, textBoxEmail.Text.Length);
        textBoxEmail.Focus();
    }
    else
    {
        // בודק סיסמה והתאמה של סיסמה
        string firstname = textBoxFirstName.Text;
        string lastname = textBoxLastName.Text;
        string email = textBoxEmail.Text;
        string password = passwordBox.Password;
        if (passwordBox.Password.Length == 0)
        {
            errorMessage.Text = "Enter password.";
            passwordBox.Focus();
        }
        else if (passwordBoxConfirm.Password.Length == 0)
        {
            errorMessage.Text = "Enter Confirm password.";
            passwordBoxConfirm.Focus();
        }
        else if (passwordBox.Password != passwordBoxConfirm.Password)
        {
            errorMessage.Text = "Confirm password must be same as password.";
            passwordBoxConfirm.Focus();
        }
        else
        {

```

```

        errormessage.Text = "";

        //כניס לדאטב בייס משתמש חדש
        OleDbConnection con_string = new
OleDbConnection(@"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\project_gal\FileServer\FileServer\FileShareDB.accdb");
        con_string.Open();

        OleDbCommand cmd = new OleDbCommand(@"INSERT INTO users([Fname],
[lname] ,[Password], [Email]) VALUES('" + firstname + "', '" + lastname + "', '" +
password + "', '" + email + "')", con_string);
        cmd.ExecuteNonQuery();
        con_string.Close();
        errormessage.Text = "You have Registered successfully.";

    }

}

}

//בודק תקינות שם ושם במשתנה את הערך
private void TextBoxFirstName_TextChanged(object sender, TextChangedEventArgs e)
{
    //
    if (textBoxFirstName.Text == "")
        errormessage.Text = "please Enter your FirstName.";
    if (!Regex.Match(textBoxFirstName.Text, "^[A-Z][a-zA-Z]*$").Success)
    {
        // first name was incorrect
        errormessage.Text="Invalid first name";
        textBoxFirstName.Focus();
        return;
    } // end if
}

//בודק תקינות שם ושם במשתנה את הערך
private void TextBoxLastName_TextChanged(object sender, TextChangedEventArgs e)
{
    if (!Regex.Match(textBoxLastName.Text, "^[A-Z][a-zA-Z]*$").Success)
    {

```

```

        // Last name was incorrect
        errorMessage.Text="Invalid last name";
        textBoxLastName.Focus();
        return;
    } // end if
}
//בדק תקינות איימיל ושם במשתנה את הערך

private void TextBoxEmail_TextChanged(object sender, TextChangedEventArgs e)
{
    if (textBoxEmail.Text.Length == 0)
    {
        errorMessage.Text = "Enter an email.";
        textBoxEmail.Focus();
    }
    else if (!Regex.IsMatch(textBoxEmail.Text,
@"^[a-zA-Z][\w\.-]*[a-zA-Z0-9]@[a-zA-Z0-9][\w\.-]*[a-zA-Z0-9]\.[a-zA-Z][a-zA-Z\.-]*[a-zA-Z]$"
))
    {
        errorMessage.Text = "Enter a valid email.";
        textBoxEmail.Select(0, textBoxEmail.Text.Length);
        textBoxEmail.Focus();
    }
    else
    {
        string email = textBoxEmail.Text;
    }
}
//בדק תקינות סיסמה ושם במשתנה את הערך

private void TextBoxPassword_TextChanged(Object sender, TextChangedEventArgs e)
{
    if ( passwordBox.Password.Length == 0&&passwordBox.Password.Length<8)
    {
        errorMessage.Text = "Enter password that has not less than 8 chars.";
        passwordBox.Focus();
    }
    else
    {
        if(!Regex.IsMatch(passwordBox.Password, @"^.* (?=.{ 8,})(?=.*\d)(?=.*[a -
z])(?=.*[A - Z])(?=.*[!*\@#$$%^&+=]).*$"))
        {
            errorMessage.Text = "Enter a valid password.";
        }
        string pass = passwordBox.Password;
    }
}

```

```

    }
}
}

```

Helper code:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Media;
using System.Windows.Media.Imaging;

namespace MediaPlayerWPF
{
    public static class Helper
    {
        public static BitmapImage MyToImage(byte[] array)
        {
            //הופך סטרים של תמונה לביטמאפ
            using (System.IO.MemoryStream ms = new System.IO.MemoryStream(array))
            {
                BitmapImage image = new BitmapImage();
                image.BeginInit();
                image.StreamSource = ms;
                image.EndInit();
                return image;
            }
        }
        //public BitmapImage ToImageLocal()
        //{
        //    byte[] imgdata = System.IO.File.ReadAllBytes("7years.png");
        //    using (System.IO.MemoryStream ms = new System.IO.MemoryStream(imgdata))
        //    {
        //        BitmapImage image = new BitmapImage();
        //        image.BeginInit();
        //        image.StreamSource = ms;
        //        image.EndInit();
        //        return image;
        //    }
        //}
    }
}

```



```

public static System.Drawing.Image MyByteArrayToImage(byte[] byteArrayIn)
{
    // הופך מערך של בייטים לאובייקט תמונה
    System.Drawing.Image returnImage = null;
    try
    {
        MemoryStream ms = new MemoryStream(byteArrayIn, 0, byteArrayIn.Length);
        ms.Write(byteArrayIn, 0, byteArrayIn.Length);
        returnImage = System.Drawing.Image.FromStream(ms, true); //Exception
occurs here
    }
    catch { }
    return returnImage;
}

public static ImageSource MyToImageSource(this System.Drawing.Image image)
{
    // מעביר לאימג לאימג סורס
    var bitmap = new BitmapImage();

    using (var stream = new MemoryStream())
    {
        image.Save(stream, System.Drawing.Imaging.ImageFormat.Png);
        stream.Position = 0;

        bitmap.BeginInit();
        bitmap.CacheOption = BitmapCacheOption.OnLoad;
        bitmap.StreamSource = stream;
        bitmap.EndInit();
    }

    return bitmap;
}
}
}

```

```
using System;
using System.Linq;
using System.Text;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Drawing;
using System.Drawing.Imaging;
using System.Windows.Forms;
using NAudio.Wave;
using System.Runtime.Serialization;
using System.Threading;

namespace FileServer
{
    class Program
    {
        // סוקט לתקשורת עם הלקוח
        static Socket socket = null;
        // לקבלת חיבורים נכנסים TCP מאזין
        static TcpListener myListener = null;
        // להמרת מחרוזות לבייטים ולהיפך ASCII קידוד
        static ASCIIEncoding asciiEnc = new ASCIIEncoding();

        static void Main(string[] args)
        {
            // והפורט של השרת IP-הגדרת כתובת ה
            IPAddress ipAddr = IPAddress.Parse("127.0.0.1");
            myListener = new TcpListener(ipAddr, 8001);
            myListener.Start(); // הפעלת המאזין
            //com = new Communicator();

            // לולאה אינסופית כדי לשמור על השרת פועל
            while (true)
            {
                try
                {
                    Console.WriteLine("the server is running at port:" +
```

```

myListener.LocalEndpoint);
    Console.WriteLine("waiting for a connection");
    socket = myListener.AcceptSocket(); // קבלת חיבור חדש
    Console.WriteLine("connection accepted");
    byte[] binDataIn = new byte[1024];
    Console.WriteLine("the message from client side");

    // עיבוד הודעות מהלקוח בלולאה
    while (true)
    {
        int k = socket.Receive(binDataIn); // קבלת נתונים מהלקוח
        string[] msgDetails = asciiEnc.GetString(binDataIn, 0,
k).Split('$');

        // טיפול בהודעות לפי סוגן
        switch (msgDetails[0])
        {
            case "1":
                HandleWavStream(msgDetails[1]); // קריאה לפונקציה
                break;
            case "2":
                HandleGallery(msgDetails[1]); // קריאה לפונקציה
                break;
        }
    }
    catch (Exception e)
    {
        // המשך הלולאה במקרה של חריגה
        continue;
    }
}

// טיפול בבקשות שקשורות לגלריה
private static void HandleGallery(string cmd)
{
    string[] fileEntries = null;
    int i = 0;
    while (cmd != "done")

```

```

    {
        byte[] binDataIn = new byte[255];

        switch (cmd)
        {
            case "names":
                fileEntries = Directory.GetFiles("img"); // קבלת רשימת
img קבצים מהתיקייה

                string songsname = String.Join("$", fileEntries.Select(x
=> x.Remove(x.IndexOf("img\\"), "img\\".Length)).ToArray());
                Console.WriteLine();

                byte[] bufferOut = asciiEnc.GetBytes(songsname); // המרת
שמות הקבצים לבייטים

                socket.Send(bufferOut, 0, bufferOut.Length,
SocketFlags.None); // שליחת הנתונים ללקוח
                break;
            case "getnext":
                Thread.Sleep(100);
                if (i == fileEntries.Length)
                    bufferOut = new byte[1];
                else
                    bufferOut = ImageToBytes(fileEntries[i++]);
                socket.Send(bufferOut, 0, bufferOut.Length,
SocketFlags.None);
                break;
        }
        int k = socket.Receive(binDataIn);
        cmd = asciiEnc.GetString(binDataIn, 0, k);
    }
}

private static byte[] ImageToBytes(string imgname)
{
    byte[] imgdata = System.IO.File.ReadAllBytes(imgname); // קריאת תוכן
הקובץ למערך בייטים
    return imgdata;
}

```

```

// טיפול WAV בבקשות שקשורות להזרמת קבצי
private static void HandleWavStream(string msg)
{
    byte[] binDataIn = new byte[1024];
    string status = "";
    System.Threading.Thread.Sleep(500); // השהיה לצורך סנכרון
    string filePath = @msg; // נתיב הקובץ מההודעה
    string outputPath = @"tempmydir\"; // ספריית הפלט
    DirectoryInfo di = Directory.CreateDirectory(outputPath); // יצירת הספרייה
    Console.WriteLine("The directory was created successfully at {0}.",
Directory.GetCreationTime(outputPath));
    int segments = SpliToSegemnts(filePath, outputPath);
    byte[] binOut = asciiEnc.GetBytes(segments.ToString());
    socket.Send(binOut);
    binOut = asciiEnc.GetBytes(string.Format("strat#{0}", segments));
    socket.Send(binOut);
    System.Threading.Thread.Sleep(200);

    // שליחת כל מקטע ללקוח
    for (int i = 1; i <= segments; i++)
    {
        string segemntPath = string.Format("{0}{1}.wav", outputPath, i);
        using (FileStream fs = new FileStream(segemntPath, FileMode.Open,
FileAccess.Read))
        {
            int length = Convert.ToInt32(fs.Length);
            binOut = new byte[length];
            fs.Read(binOut, 0, length); // קריאת תוכן הקובץ למערך בייטים
            fs.Close();
            socket.Send(binOut); // שליחת המקטע ללקוח
            binOut = asciiEnc.GetBytes("end");
            System.Threading.Thread.Sleep(1500);
            socket.Send(binOut); // שליחת הודעה על סיום המקטע
            int x = socket.Receive(binDataIn); // קבלת אישור מהלקוח
            status = asciiEnc.GetString(binDataIn, 0, x);
            Console.WriteLine(status);
        }
    }
    // מחיקת קבצים זמניים
    string[] files = Directory.GetFiles(@"tempmydir");
    foreach (string file in files)
        File.Delete(file);
}

```

```

}

// למקטעים WAV חלוקת קובץ
public static int SplitToSegments(string filePath, string outputPath)
{
    using (var wfr = new WaveFileReader(filePath))
    {
        TimeSpan totalTime = wfr.TotalTime; // משך זמן הקובץ
        var extra = totalTime.TotalMilliseconds % 1000; // חישוב הזמן הנוסף
        double num = totalTime.TotalMilliseconds;
        Console.WriteLine(num);
        int count = 2; int segment = 1;
        double start = 0, end = count * 1000;

        // חלוקת הקובץ לפי משך הזמן
        while (end < num)
        {
            WavFileUtils.TrimWavFile(filePath, outputPath,
TimeSpan.FromMilliseconds(start), TimeSpan.FromMilliseconds(end), segment);
            start = end;
            count += 2;
            end = count * 1000;
            segment++;
        }
        WavFileUtils.TrimWavFile(filePath, outputPath,
TimeSpan.FromMilliseconds(start), TimeSpan.FromMilliseconds(num), segment);
        return segment;
    }
}

// הדפסת מערך בייטים (לצורך ניפוי שגיאות)
public static void PrintArr(byte[] arr)
{
    for (int i = 0; i < arr.Length; i++)
    {
        Console.Write(arr[i]);
    }
}

```

File server WavFileUtils code:

```
using NAudio.Wave;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FileServer
{
    class WavFileUtils
    {
        // פונקציה לקיצוץ קובץ WAV
        public static void TrimWavFile(string inPath, string outputPath, TimeSpan
cutFromStart, TimeSpan cutFromEnd, int segment)
        {
            outputPath += segment + ".wav"; // הוספת שם המקטע לקובץ הפלט
            using (WaveFileReader reader = new WaveFileReader(inPath)) // קריאת קובץ
המקורי WAV-ה
            {
                using (WaveFileWriter writer = new WaveFileWriter(outputPath,
reader.WaveFormat))
                {
                    int bytesPerMillisecond = reader.WaveFormat.AverageBytesPerSecond /
1000; // חישוב מספר הבייטים במילישנייה

                    int startPos = (int)cutFromStart.TotalMilliseconds *
bytesPerMillisecond; // חישוב מיקום ההתחלה
                    startPos = startPos - startPos % reader.WaveFormat.BlockAlign; //
התאמת מיקום ההתחלה ליישור הבלוק
                    int endPos = (int)cutFromEnd.TotalMilliseconds *
bytesPerMillisecond; // חישוב מיקום הסיום
                    endPos = endPos - endPos % reader.WaveFormat.BlockAlign; // התאמת
מיקום הסיום ליישור הבלוק

                    //int endBytes = (int)cutFromEnd.TotalMilliseconds *
bytesPerMillisecond;
                    //endBytes = endBytes - endBytes % reader.WaveFormat.BlockAlign;
                    //int endPos = (int)reader.Length - endBytes;

                    //קריאה לפונקציה שמבצעת את החיתוך בפועל/
                    TrimWavFile(reader, writer, startPos, endPos);
                }
            }
        }
    }
}
```


ביבליוגרפיה

- Streaming - [video Streaming: Concepts, Algorithms, and Systems](#)
- threads - [Thread \(computing\) wikipedia](#)
- TCP/IP - [TCP/IP wikipedia](#)