תמונה שמכילה גופן, גרפיקה, ורוד, מג'נטה

התיאור נוצר באופן אוטומטי

**שם התלמיד** – עופר איל מרמרי

**ת"ז** – 326090362

**התמחות** - סייבר

**בית ספר** - תיכון כרמים בנימינה

**מנחים/מורים מלווים** - מרי גבע

**תאריך הגשה** – 28/5/2023

תוכן עניינים

[Umziboom - ייזום 3](#_Toc135594963)

[תיאור ראשוני של המערכת 3](#_Toc135594964)

[הגדרת הלקוח 3](#_Toc135594965)

[הגדרת יעדים/מטרות 3](#_Toc135594966)

[בעיות, תועלות וחסכונות 3](#_Toc135594967)

[האם צפויים קשיים או מגבלות בהגדרת המערכת 3](#_Toc135594968)

[תיחום הפרויקט 4](#_Toc135594969)

[Umziboom - אפיון 5](#_Toc135594970)

[פרוט המערכת: 5](#_Toc135594971)

[מה היכולות שהיא תעניק למשתמש: 5](#_Toc135594972)

[פירוט הבדיקות 6](#_Toc135594973)

[תכנון לוח זמנים לפרויקט 7](#_Toc135594974)

[ניהול סיכונים בפרויקט 8](#_Toc135594975)

[Umziboom - מסמך ניתוח 9](#_Toc135594976)

[יכולות בצד לקוח: 9](#_Toc135594977)

[יכולות בצד שרת: 15](#_Toc135594978)

[Umziboom - העיצוב 17](#_Toc135594979)

[תיאור הארכיטקטורה של המערכת המוצעת 17](#_Toc135594980)

[תיאור החומרה: 17](#_Toc135594981)

[רכיבי המערכת: 17](#_Toc135594982)

[תיאור הטכנולוגיה הרלוונטית 18](#_Toc135594983)

[תיאור מודולים בהם נעשה שימוש 19](#_Toc135594984)

[מודולים מיובאים: 23](#_Toc135594985)

[מודולים שלי: 23](#_Toc135594986)

[צד לקוח: 23](#_Toc135594987)

[צד שרת 29](#_Toc135594988)

[תיאור סביבת הפיתוח 38](#_Toc135594989)

[כלי פיתוח: 38](#_Toc135594990)

[כלי בדיקה: 38](#_Toc135594991)

[תיאור האלגוריתמים המרכזיים בפרויקט 39](#_Toc135594992)

[בעיה 1. איך תתבצע התקשורת? 39](#_Toc135594993)

[בעיה 2. איך תתבצע העבודה במקביל? 39](#_Toc135594994)

[בעיה 3. מה תהיה שיטת ההצפנה? 40](#_Toc135594995)

[בעיה 4. איך תתבצע השמעת המוזיקה? 40](#_Toc135594996)

[תיאור מסכי הפרויקט: 41](#_Toc135594997)

[מסך כניסה התחברות 41](#_Toc135594998)

[מסך יצירת משתמש 42](#_Toc135594999)

[מסך בית 42](#_Toc135595000)

[מסך יצירת קובץ 43](#_Toc135595001)

[מסך עריכת קובץ 43](#_Toc135595002)

[עריכת רצועה 44](#_Toc135595003)

[הוספת רצועה 44](#_Toc135595004)

[תיאור פרוטוקול התקשורת 45](#_Toc135595005)

[צד לקוח: 45](#_Toc135595006)

[פרוטוקול שליחת קובץ שרת ולקוח 45](#_Toc135595007)

[צד שרת: 46](#_Toc135595008)

[תיאור מבני הנתונים 48](#_Toc135595009)

[מסד נתונים: 48](#_Toc135595010)

[סקירת חולשות והאיומים 49](#_Toc135595011)

[שכבת האפליקציה: 49](#_Toc135595012)

[שכבת התעבורה: 49](#_Toc135595013)

[הפעלת המערכת 49](#_Toc135595014)

[Umziboom – הקוד 50](#_Toc135595015)

[עבודה במקביל: 50](#_Toc135595016)

[החלפת מפתחות דיפי הלמן: 51](#_Toc135595017)

[חיבור שכבות מוזיקה: 52](#_Toc135595018)

[שליחת קובץ עריכה: 53](#_Toc135595019)

[אתחול רצועות בשרת עריכה: 54](#_Toc135595020)

[התחלת עריכת קובץ עריכה משותף: 55](#_Toc135595021)

[קליטת קובץ בחלקים: 56](#_Toc135595022)

[Umziboom – בדיקות 57](#_Toc135595023)

[פירוט הבדיקות 57](#_Toc135595024)

[Umziboom – מדריך למשתמש 61](#_Toc135595025)

[התחברות למערכת: 61](#_Toc135595026)

[הרשמה למערכת: 62](#_Toc135595027)

[מסך בית: 63](#_Toc135595028)

[מסך יצירת קובץ עריכה משותף: 64](#_Toc135595029)

[מסך עריכת קובץ עריכה משותף: 65](#_Toc135595030)

[מסך עריכת קובץ עריכה משותף: 66](#_Toc135595031)

[העלאת רצועה: 67](#_Toc135595032)

[Umziboom – רפלקציה 68](#_Toc135595033)

[Umziboom – ביבליוגרפיה 69](#_Toc135595034)

[נספח א – החקר שבוצע בפרויקט 70](#_Toc135595035)

[client\_comm.py 70](#_Toc135595036)

[client\_Logic.py 73](#_Toc135595037)

[client\_protocol.py 78](#_Toc135595038)

[data\_base.py 81](#_Toc135595039)

[edit\_group.py 84](#_Toc135595040)

[edit\_group\_com.py 91](#_Toc135595041)

[encrypt.py 93](#_Toc135595042)

[gui.py 95](#_Toc135595043)

[music\_strip.py 115](#_Toc135595044)

[server\_com.py 117](#_Toc135595045)

[server\_logic.py 121](#_Toc135595046)

[server\_protocol.py 125](#_Toc135595047)

# Umziboom - ייזום

## תיאור ראשוני של המערכת

הפרויקט שלי עוסק במוזיקה. המערכת היא תוכנת עריכה של מוזיקה המאפשרת לערוך קטעי מוזיקה שונים, בתוכנה יהיה אפשר לערוך כמה רצועות מוזיקה ולחבר אותן לקובץ אחד. העריכה של המוזיקה תתבצע בצורה שיתופית מרחוק כך שכמה אנשים יוכלו לערוך ביחד את אותו קטע מוזיקה כשכל אחד עורך ממחשב אחר. המוצר המוגמר יאפשר לאנשים לערוך מוזיקה מרחוק עם החברים שלהם, תהיה להם את האפשרות להכניס קטעי קול לתוכנה בתור רצועות ואת כל הרצועות יהיה אפשר לערוך בנפרד כך שבסוף יהיה אפשר לייצר קובץ מוזיקה שמורכב מכמה קבצים ערוכים. אני אוהב מוזיקה ומנגן בגיטרה והסיבה שבחרתי בפרויקט הזה הוא בגלל שרציתי להתנסות בנושא של עריכת מוזיקה וגם ליצור מוצר שכרגע לא קיים בשוק. האתגרים שצפויים לי הם למידה על קבצי מוזיקה ואיך אני יכול לערוך אותם, בנוסף יש את האתגר של בניית ממשק המשתמש וגם בניית שרת שיוכל לחבר בין הלקוחות.

## הגדרת הלקוח

המערכת מיועדת לכל בן אדם או מוזיקאי שרוצה לערוך מוזיקה עם כמה אנשים מבלי הצורך להיפגש פיזית עם האנשים כדי לערוך איתם מוזיקה.

## הגדרת יעדים/מטרות

המטרה המרכזית של המערכת היא לאפשר לאנשים לערוך מוזיקה עם החברים שלהם מרחוק. בעריכה של קובץ המוזיקה הלקוח יוכל להוסיף רצועות מוזיקה נוספות, ולשלוט באורך ובמיקום שלהן. בסופו של דבר יהיה ללקוח קובץ מוזיקה ערוך שמורכב מהרצועות שהוא והחברים שלו הוסיפו וערכו.

## בעיות, תועלות וחסכונות

הבעיה כיום היא שבשוק נוצרות יותר ויותר מערכות המאפשרות לבצע עבודה מרחוק אבל אף אחת מהן לא מאפשרת עריכה מרחוק של מוזיקה. התועלת של התוכנה שלי היא שהיא תענה על הצורך הזה ותאפשר לאנשים לערוך מוזיקה מרחוק. המערכת תיתן שירותים של העלאת קבצי מוזיקה לקובץ עריכה משותף שיאפשר לכמה אנשים לעבוד עליו בו זמנית ובסוף יהיה אפשר גם להוריד למחשב את הקובץ הערוך. כרגע בשוק יש מספר רב של מוצרים המאפשרים לערוך מוזיקה אבל אף אחד מהם לא עונה על הצורך של עריכת מוזיקה מרחוק ,לכן המערכת שלי תהיה הראשונה מסוגה.

## האם צפויים קשיים או מגבלות בהגדרת המערכת

הקושי הגדול בהגדרת המערכת הוא ללמוד עריכת קטעי מוזיקה ולהצליח לשלב בין תוכנת העריכה לשרת ולקשר בין הלקוחות. בנוסף יש קושי של שמירת הנתונים של קבצי העריכה ברגע שמפסיקים לערוך אותם כדי שהלקוחות יוכלו לחזור ולערוך את הקובץ מאותה נקודה.

## תיחום הפרויקט

הפרויקט הולך לעסוק בנושא של רשתות, בפרויקט הולך להיות שרת שמחבר בין כמה לקוחות. בנוסף הפרויקט יעסוק גם בהצפנה(המידע על הסיסמאות והשמות משתמש יהיה מוצפן). הפרויקט יעסוק גם בבסיסי נתונים שישמרו נתונים על המשתמשים ועל קבצי העריכה של המוזיקה.

# Umziboom - אפיון

## פרוט המערכת:

המערכת היא מערכת עריכת מוזיקה שיתופית, במערכת יהיו מספר קבצי עריכה משותפים של קבוצות שונות וכל קבוצה תוכל לערוך במשותף את הקובץ מוזיקה שלה. לכל משתמש תהיה אפשרות להוסיף ולהסיר רצועות מוזיקה שונות לקובץ עריכה, בנוסף תהיה גם אפשרות לערוך את מיקום הרצועה האורך שלה והווליום שלה. השרת ידאג לעדכן את כל השינויים בקובץ מוזיקה המשותף אצל כל חברי הקבוצה. בכל שלב של עריכה תהיה למשתמש אפשרות להפסיק לערוך את הקובץ והמערכת תשמור את השינוי האחרון כדי שהמשתמש יוכל לגשת אח"כ לקובץ ולהתחיל לערוך מאותו מקום שהוא עצר, וגם המשתמש יוכל בכל רגע לבקש לשמור את קובץ המוזיקה(קובץ המורכב משילוב של כל הרצועות) למחשב האישי שלו. למשתמשים תהיה גם אפשרות ללחוץ על כפתור undo כדי להחזיר או לבטל שינויים שנעשו בקובץ המשותף.

## מה היכולות שהיא תעניק למשתמש:

יצירת משתמש – כל מחשב שיכנס למערכת יוכל להוסיף משתמש חדש שלו למערכת עם שם וסיסמה משלו

התחברות למערכת – כל משתמש יוכל להתחבר למערכת עם הסיסמה והשם משתמש שלו

יצירת קובץ עריכה – המשתמש יוכל ליצור קובץ עריכה משותף, לתת שם לקובץ ולבחור את הקבוצה של האנשים שאיתם הוא רוצה לעבוד מתוך רשימת המשתמשים הקיימים במערכת.

התחלת עבודה על קובץ עריכה – למשתמש תהיה רשימה של קבצי העריכה המשותפים שלו והוא יוכל לבחור את הקובץ עליו הוא רוצה לעבוד ולהתחיל לערוך אותו.

עריכת רצועות מוזיקה בקובץ – לכל משתמש תהיה אפשרות לבחור רצועה שלא עובדים עליה כרגע ולהתחיל לערוך אותה. בעריכה הוא יוכל לשנות את מיקום הרצועה בציר הזמן, הוא יוכל גם לשנות גם את אורך הרצועה ולקצר אותה, וגם הוא יוכל לשנות את הווליום שלה. המשתמש יוכל גם ללחוץ על כפתור ההundo כדי להחזיר שינויים שנעשו ברצועה.

הוספה והסרה של רצועות מוזיקה בקובץ – כל משתמש יוכל לבחור איזה רצועות בקובץ הוא רוצה להסיר ולהסיר אותן והוא גם יוכל להעלות קבצי מוזיקה בפורמט mp3 או wav בתור רצועות לקובץ העריכה המשותף.

לתת שם(תווית) לרצועות מוזיקה – בעת העלאת קבצי המוזיקה בתור רצועות המשתמש יוכל לתת שם לרצועה כדי שיוכל לזהות אותה בקובץ העריכה המשותף.

שמירת קובץ למחשב – כל משתמש יוכל בכל רגע נתון להחליט שהוא רוצה לשמור את קובץ העריכה המשותף(קובץ המוזיקה שמורכב משילוב של כל הרצועות) על מחשבו בתור קובץ mp3 או .wav

השמעת מוזיקה – כל משתמש שעובד על קובץ המוזיקה השיתופי יוכל ללחוץ על כפתור שישמיע לו את השילוב של כל רצועות המוזיקה, הוא יוכל לבחור גם להשתיק את חלק מהרצועות.

## פירוט הבדיקות

| **מספר** | **שם הבדיקה (שם שיעיד על תוכן הבדיקה)** | **מה אמורה לבדוק** | **איך מתכננים לבדוק**  **(לתאר בפירוט את שלבי הבדיקה)** |
| --- | --- | --- | --- |
| 1 | בדיקת יציבות שרת | לבדוק שהשרת יציב והוא לא נופל בשום שלב | מתחברים לשרת עם כמות מרובה של משתמשים ומנסים להתנתק בשלבים שונים בריצה במערכת |
| 2 | בדיקת עריכה של רצועות מוזיקה | לבדוק שאכן יכולת העריכה עובדת | לנסות לערוך רצועת מוזיקה, לשנות את האורך שלה, לשנות ווליום, לשנות את המיקום שלה על ציר הזמן |
| 3 | בדיקת התחברות לשרת | לראות שאכן ניתן ליצור התחברות בין לקוח לשרת | לנסות להתחבר לשרת דרך הלקוח ולראות שהוא מקבל את השם משתמש והסיסמה |
| 4 | הוספת משתמש | לראות שאכן ניתן ליצור משתמש חדש | לנסות ליצור משתמש חדש עם שם משתמש וסיסמה ולראות שהוא אכן מתווסף למערכת |
| 5 | יצירת קובץ עריכה משותף | לראות שברגע שיוצרים קובץ עריכה חדש הוא מתווסף גם אצל שאר חברי הקבוצה | לנסות ליצור קובץ עריכה משותף ולבדוק שהוא מתווסף גם אצל שאר חברי הקבוצה |
| 6 | סנכרון שינויים | לראות שהשינויים שאתה מבצע בקובץ המוזיקה מתבצעים גם אצל שאר חברי הקבוצה | לנסות לבצע שינויים שונים בקובץ עריכה המשותף ולבדוק שכל השינויים מתבצעים גם אצל כל שאר חברי הקבוצה |
| 7 | בדיקת undo | לבדוק שיכולת undo עובדת | לנסות לבצע undo על רצועה בקובץ עריכה ולראות שאכן הרצועה חוזרת למצב הקודם |
| 8 | הוספה והסרה של רצועות מוזיקה | לבדוק שאכן ניתן להוסיף ולהסיר רצועות מוזיקה | לנסות להוסיף או להסיר רצועות מוזיקה בקובץ העריכה המשותף |
| 9 | השמעת קובץ מוזיקה | לבדוק שההשמעה של קובץ המוזיקה עובדת | לנסות להשמיע את קובץ המוזיקה המשותף אצל הלקוח ולנסות להשתיק חלק מהרצועות ואז להשמיע את הקובץ |
| 10 | שמירת קובץ מוזיקה אצל מחשב לקוח | לבדוק שניתן לשמור את קובץ המוזיקה(שילוב של כל הרצועות) אצל מחשב הלקוח | לנסות לשמור את קובץ העריכה אצל מחשב הלקוח ולראות שהוא נשמר על מחשב הלקוח |
| 11 | זמני תגובה | לבדוק זמני תגובה של המערכת כאשר יש מספר שונה של משתמשים | ליצור קבוצות עם מספר שונה של אנשים או לעבוד כמה קבוצות בו זמנית ולבדוק אם זה משפיע על מהירות זמני התגובה של המערכת ואם נוצר עומס בשרת |
| 13 | אבטחה | לבדוק שרק חברים שהוזמנו יכולים לערוך את הקובץ המשותף | לראות שלאף משתמש אין גישה לקובץ עריכה משותף שהוא לא חלק ממנו |
| 14 | שלמות | לבדוק שהמידע שכל חבר בקבוצת העריכה רואה שלם וזהה לשאר החברים | לנסות לבצע מספר שינויים ועריכות על רצועות המוזיקה בקובץ העריכה המשותף ולבדוק שהשינויים מתבצעים במדויק אצל שאר חברי הקבוצה |

## תכנון לוח זמנים לפרויקט

| **פעילות** | **זמן התחלה מתוכנן** | **זמן סיום מתוכנן** | **זמן התחלה בפועל** | **זמן סיום בפועל** | **הערות** |
| --- | --- | --- | --- | --- | --- |
| יזום | 25/10/22 | 1/11/22 | 25/10/22 | 1/11/22 |  |
| אפיון | 10/11/22 | 15/11/22 | 16/11/22 | 19/12/22 |  |
| ניתוח | 6/11/22 | 10/12/22 | 19/12/22 | 1/1/23 |  |
| עיצוב | 10/12/22 | 20/1/23 | 1/1/23 | 15/1/23 |  |
| קוד | 20/1/23 | 1/3/23 | 15/1/23 | 20/1/2023 |  |
| עבודה על יכולות עריכת מוזיקה | 20/1/23 | 21/1/23 | 15/1/23 | 29/1/2023 |  |
| בניית ממשק משתמש לקוח | 21/1/23 | 28/1/23 | 29/1/2023 | 5/2/2023 |  |
| בניית ממשק משתמש כולל גרפיקה של תוכנת העריכה | 28/1/23 | 11/2/23 | 5/2/2023 | 5/3/2023 |  |
| בניית שרת התומך בהתחברות לקוחות ויצירת קבוצות | 11/2/23 | 18/2/23 | 5/3/2023 | 5/4/2023 |  |
| השלמת שאר הפונקציונליות הנדרשת | 25/2/23 | 28/2/23 | 5/4/2023 | 19/4/2023 |  |
| גרסה ראשונית | /1/2328 | 1/3/23 | 19/4/2023 | 26/4/2023 |  |
| מסמך בדיקות | 1/4/23 | 20/4/23 | 26/4/2023 | 30/4/2023 |  |
| מדריך למשתמש | 20/4/23 | 30/4/23 | 30/4/2023 | 2/5/2023 |  |
| גרסה סופית | 30/4/23 | 1/5/23 | 2/5/23 | 12/5/2023 |  |
| סגירת תיק פרויקט | 30/4/23 | 1/5/23 | 12/5/2023 | 20/5/2023 |  |
| מצגת הסבר | 1/5/23 | 8/5/23 | 20/5/2023 | 22/5/2320 |  |

## ניהול סיכונים בפרויקט

| **הסיכון** | **פירוט הסיכון** | **רמת הסיכון** | **תיאור דרכים להתמודדות עם הסיכון ולהקטין אותו** | **מה בוצע בפועל** | **תאריך** |
| --- | --- | --- | --- | --- | --- |
| אי עמידה בזמנים | פרויקט לא יושלם | קשה | * הקדמת לו"זים משימות * ארגון הזמן בצורה יעילה * להתחיל בדברים היותר קשים | סידור הזמן שלי לא בוצע בצורה המיטבית ואיחרתי חלק מתאריכי ההגשה | 21/5/2023 |
| שרת לא יציב | פרויקט לא תומך בעבודה מרובת משתמים | קשה | * תכנון מראש * ביצוע בדיקות כדי לאשר את יציבות השרת | כתבתי את השרת בצורה בטוחה ועשיתי כמה בדיקות כדי לבטח את יציבותו והוא אכן יציב | 21/5/2023 |
| סנכרון שינויים בין חברי הקבוצה לא עובד | פרויקט לא יכול להתקיים בגלל שהסנכרון לא עובד | קשה | תכנון מראש של אלגוריתם הסנכרון | תכננתי את הפרויקט צורה שהסנכרון יעבוד בלי בעיות והסנכרון עובד בלי בעיות | 21/5/2023 |
| ממשק משתמש לא נוח | עריכת קבצי המוזיקה בפרויקט לא נוחה במיוחד | קל | תכנון מראש של מבנה ממשק המשתמש ובניה נכונה | תכננתי את ממשק המשתמש בצורה שהיא נוחה ויעילה לשימוש | 21/5/2023 |
| חלק מיכולות עריכת המוזיקה לא עובדות | המשתמש לא יוכל להשתמש בחלק מיכולות העריכה | בינוני | בדיקה מראש של יכולות עריכת המוזיקה ולוודא שכולן אכן עובדות | בדקתי איזה יכולות עריכת מוזיקה עובדות ומימשתי רק את אלה שעבדו | 21/5/2023 |

# Umziboom - מסמך ניתוח

## יכולות בצד לקוח:

שם היכולת: הרשמה למערכת

מהות היכולת: רישום משתמש חדש במערכת (קליטת סיסמא ושם משתמש)

אוסף יכולות:

* ממשק משתמש – מסך הרשמה
* קליטת נתונים
* בדיקת תקינות
* הצפנה
* שליחה לשרת
* קבלת תשובה מהשרת
* פענוח
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת

שם היכולת: התחברות למערכת

מהות היכולת: התחברות משתמש קיים למערכת(קליטת סיסמא ושם משתמש)

אוסף יכולות:

* ממשק משתמש – מסך התחברות
* קליטת נתונים
* בדיקת תקינות
* הצפנה
* שליחה לשרת
* קבלת תשובה מהשרת
* פענוח
* הצגת התשובה למשתמש

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת

שם היכולת: יצירת קובץ עריכה

מהות היכולת: לבחור חברי קבוצה ולפתוח איתם קובץ עריכה משותף

אוסף יכולות:

* בחירת חברי קבוצה
* ממשק משתמש – מסך יצירת קובץ
* קליטת נתונים לשם הקובץ
* בדיקת תקינות נתונים
* הצפנה
* שליחה לשרת
* קבלת תשובה מהשרת על תקינות שם הקובץ
* פענוח
* הצגת התשובה למשתמש
* הוספת קובץ בממשק משתמש – מסך הבית

רשימת אובייקטים: ממשק משתמש, תקשורת, הצפנה/פיענוח, רשימת חברים, קובץ

שם היכולת: עריכת רצועות מוזיקה בקובץ

מהות היכולת: עריכת רצועת מוזיקה בקובץ עריכה משותף(שינוי ווליום, אורך, מיקום)

אוסף יכולות:

* ממשק משתמש – עריכת קובץ
* שינוי מיקום הרצועה על ציר הזמן
* שינוי ווליום רצועה
* שינוי אורך רצועה
* שכפול רצועה
* UNDO(ביטול שינוי שנעשה ברצועה)
* הצפנה
* שליחת שינוי לשרת
* קבלת תשובה מהשרת
* פענוח

רשימת אובייקטים: ממשק משתמש, תקשורת, הצפנה/פיענוח, עריכה, קובץ

שם היכולת: התחלת עבודה על קובץ עריכה

מהות היכולת: בחירת קובץ עריכה משותף מרשימת הקבצים הקיימים ולהתחיל לעבוד עליו

אוסף יכולות:

* ממשק משתמש - מסך בית(רשימת קבצים)
* בחירת קובץ
* הצפנה
* שליחת הודעה לשרת על תחילת עבודה על קובץ
* קבלת תשובה מהשרת
* קבלת קובץ משרת
* פענוח

רשימת אובייקטים: ממשק משתמש, תקשורת, הצפנה/פיענוח, קובץ

שם היכולת: הוספה והסרה של רצועות מוזיקה בקובץ

מהות היכולת: למחוק ולהוסיף רצועות מוזיקה לקובץ העריכה המשותף

אוסף יכולות:

* ממשק משתמש – עריכת קובץ
* בחירת רצועה להסרה
* בחירת רצועה להוספה מקבצי המחשב(אם אפשרות להעלות או WAV או MP3)
* הצפנה
* שליחת שינוי לשרת

רשימת אובייקטים: ממשק משתמש, תקשורת, עריכה, הצפנה/פיענוח, קובץ

שם היכולת: לתת שם(תווית) לרצועות מוזיקה

מהות היכולת: לתת שם לרצועת מוזיקה בעת העלאה כדי לדעת להבדיל בין הרצועות השונות

אוסף יכולות:

* ממשק משתמש - עריכה
* קבלת קלט ממשתמש
* בדיקת קלט
* הצפנה
* שליחת שם לשרת
* קבלת תשובה מהשרת
* פענוח

רשימת אובייקטים: ממשק משתמש, תקשורת, הצפנה/פיענוח

שם היכולת: שמירת קובץ למחשב

מהות היכולת: לשמור את התוצר הסופי של קובץ העריכה(שילוב של כל הרצועות ביחד)

אוסף יכולות:

* ממשק משתמש – עריכה
* נתינת שם לקובץ
* בחירת תיקיה לשמירה במחשב
* בחירה אם לשמור בתור WAV או MP3

רשימת אובייקטים: ממשק משתמש, עריכה

שם היכולת: השמעת מוזיקה

מהות היכולת: לשמוע את קובץ המוזיקה(שילוב של כל הרצועות)

אוסף יכולות:

* ממשק משתמש – עריכה
* השתקת רצועות
* הוצעת רצועות מהשתק
* השמעת קובץ מוזיקה(שילוב של כל הרצועות)

רשימת אובייקטים: ממשק משתמש, עריכה, נגן מוסיקה

## יכולות בצד שרת:

שם היכולת: הרשמה למערכת

מהות היכולת: רישום משתמש חדש במערכת

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* בדיקה מול בסיס הנתונים
* גיבוב סיסמא
* הוספה לבסיס נתונים
* החזרת תשובה מוצפנת

רשימת אובייקטים: הצפנה/פענוח, תקשורת, בסיס נתונים

שם היכולת: התחברות למערכת

מהות היכולת: אישור כניסה למערכת למשתמש קיים

אוסף יכולות:

* קבלת נתונים מהלקוח
* פענוח
* בדיקה מול בסיס הנתונים
* החזרת תשובה מוצפנת

רשימת אובייקטים: הצפנה/פענוח, תקשורת, בסיס נתונים

שם היכולת: יצירת קובץ עריכה

מהות היכולת: יצירת קובץ עריכה משותף ללקוח

אוסף יכולות:

* קבלת נתונים מהלקוח(איזה חברי קבוצה ושם הקובץ)
* פענוח
* בדיקה מול בסיס הנתונים(ששם הקובץ לא קיים עבור משתמש זה)
* יצירת קובץ עריכה משותף(אם אין קובץ עם אותו שם)
* החזרת תשובה מוצפנת

רשימת אובייקטים: הצפנה/פענוח, תקשורת, בסיס נתונים, קובץ מוזיקה

שם היכולת: עריכת רצועות מוזיקה בקובץ

מהות היכולת: עדכון שינויים בקבצי מוזיקה משותפים אצל השרת והלקוחות

אוסף יכולות:

* קבלת נתונים מהלקוח(פעולה שבוצעה על הקובץ)
* פענוח
* בדיקת תקינות הפעולה
* גישה לבסיס נתונים להחזרת מידע(אם התבצע UNDO או REDU)
* עדכון קובץ מוזיקה
* החזרת תשובה מוצפנת ללקוח
* שליחת נתונים לשאר הלקוחות(חברי הקבוצה)

רשימת אובייקטים: הצפנה/פענוח, תקשורת, בסיס נתונים, קובץ מוזיקה

שם היכולת: התחלת עבודה על קובץ עריכה

מהות היכולת: לספק ללקוח את קובץ העריכה המשותף שהוא רוצה לעבוד עליו

אוסף יכולות:

* קבלת נתונים מהלקוח(שם הקובץ עליו הוא רוצה לעבוד)
* פענוח
* בדיקת תקינות נתונים(לראות שללקוח אכן יש גישה לקובץ)
* החזרת תשובה מוצפנת ללקוח
* שליחת קובץ מוצפן ללקוח

רשימת אובייקטים: הצפנה/פענוח, תקשורת, קובץ מוזיקה

שם היכולת: הוספה והסרה של רצועות מוזיקה בקובץ

מהות היכולת: עדכון שינויים בקבצי מוזיקה משותפים אצל השרת והלקוחות

אוסף יכולות:

* קבלת נתונים מהלקוח(רצועה להסיר או להוסיף)
* פענוח
* בדיקת תקינות הפעולה
* החזרת תשובה מוצפנת ללקוח
* עדכון קובץ מוזיקה
* שליחת נתונים מוצפנים לשאר הלקוחות(חברי הקבוצה)

רשימת אובייקטים: הצפנה/פענוח, תקשורת, קובץ מוזיקה

# Umziboom - העיצוב

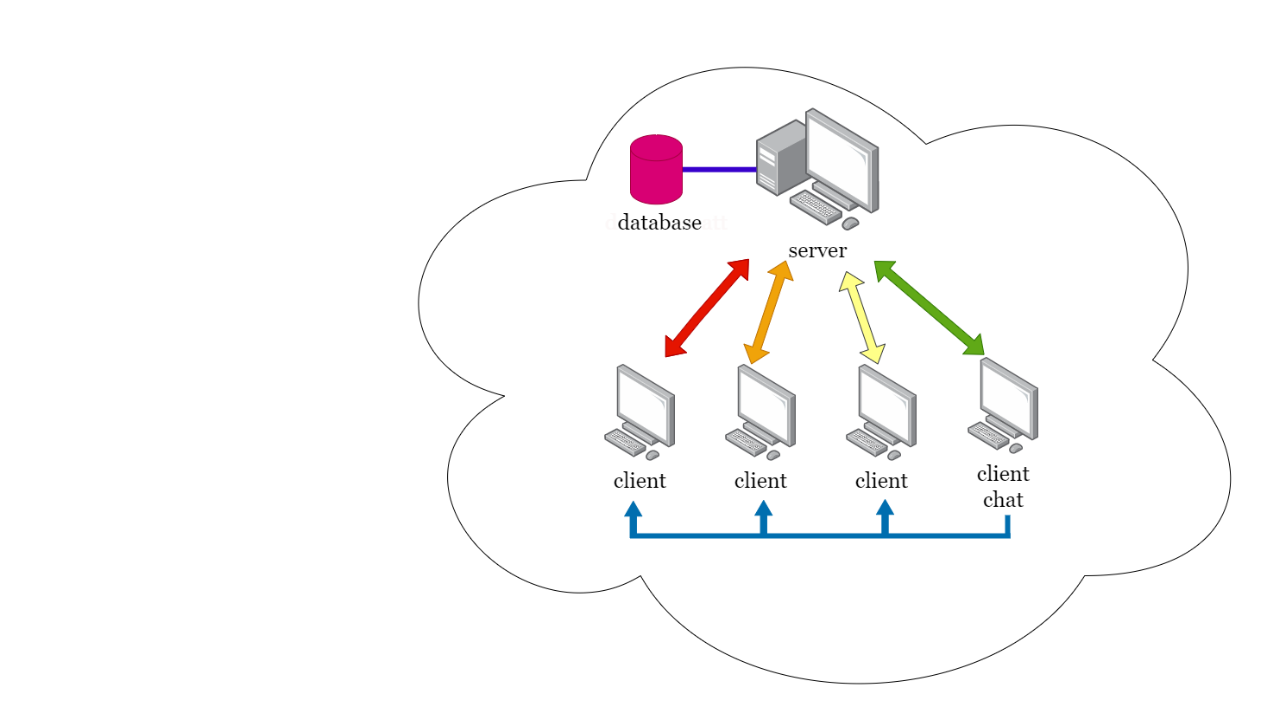
## תיאור הארכיטקטורה של המערכת המוצעת

### תיאור החומרה:

* שרת מבוסס Windows
* לקוחות מבוססי Windows
* מסד נתונים
* מערכת קבצים

### רכיבי המערכת:

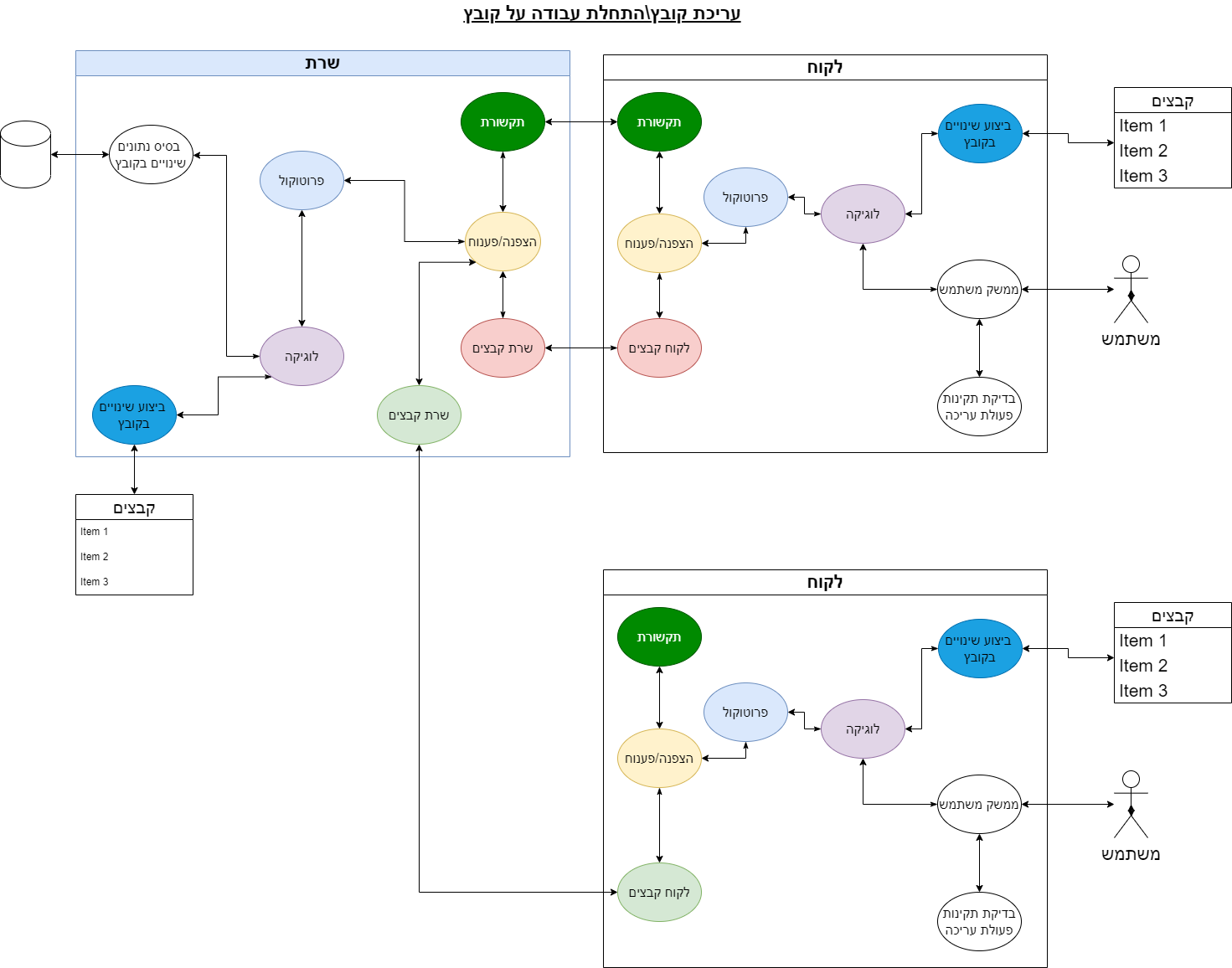
* לקוחות
* שרתים
* אחסון פנימי – file system
* אחסון חיצוני – מסד נתונים אשר נמצא בשרת – SQL

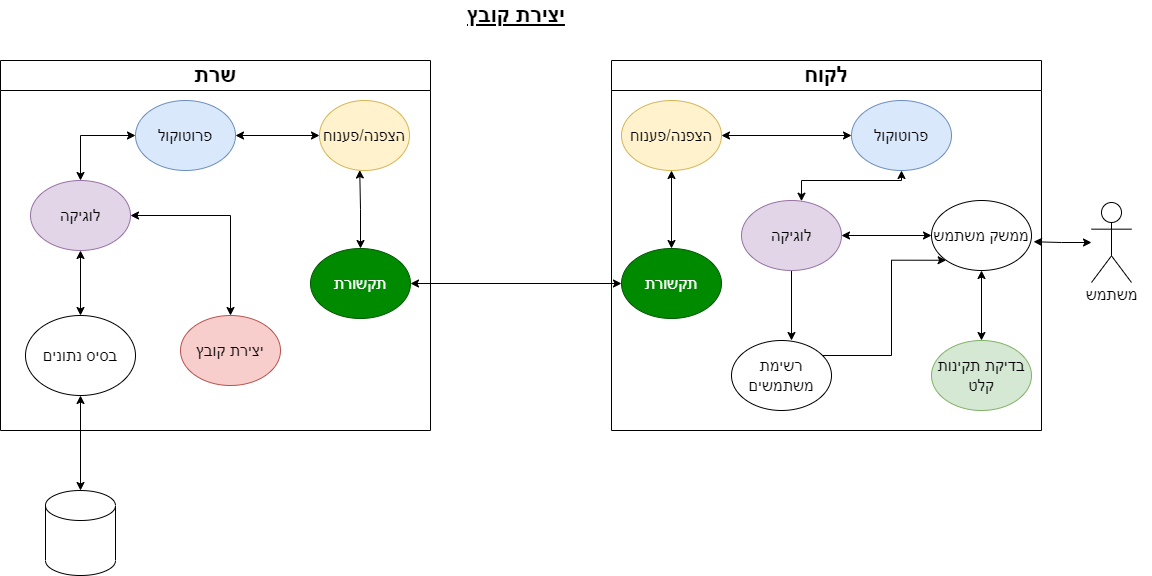
תפקיד השרת הוא להעביר את רוב התקשורת בין הלקוחות בהעברה של קבצים והוראות. השרת אחראי גם על שמירת הקבצים. תקשורת הצ'אט תתבצע אצל הלקוחות בינם לבין עצמם.

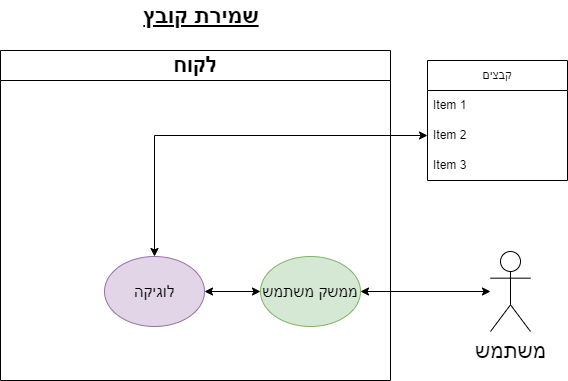
## תיאור הטכנולוגיה הרלוונטית

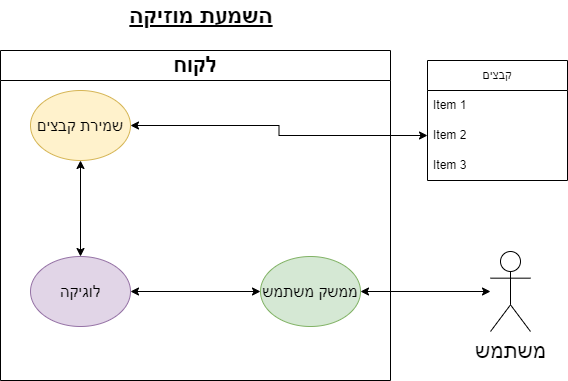
* Python: למימוש המערכת קוד שרת קוד לקוח וממשק משתמש
* מערכת הפעלה: windows
* תקשורת: מתבצעת בעזרת socket –ים
* בסיס נתונים: באמצעות שפת SQL
* הצפנה: כדי לשמור על הנתונים המועברים בטוחים
* Threading: כדי לבצע פעולות בו זמנית כמו תקשורת קבלת קלט ממשתמש וכו...
* שרת מרובה לקוחות: מאפשר לבצע חיבור ועבודה עם מספר לקוחות ומנהל עבודה.
* Hashing: גיבוב סיסמאות אצל השרת והלקוח כדי לשמור על פרטיות וביטחון.

## תיאור מודולים בהם נעשה שימוש









## מודולים מיובאים:

|  |  |
| --- | --- |
| **שם המודול** | **תיאור** |
| socket | כדי לבצע את התקשורת |
| select | כדי לבצע תקשורת של שרת מרובה לקוחות |
| Sqlite3 | כדי לעבוד עם מסד הנתונים |
| time | כדי לגשת לזמן ולבדוק דברים שונים |
| threading | כדי לבצע כמה משימות בו זמנית מבלי לתקוע את המערכת |
| queue | עצם התור |
| wxpython | מחלקה למימוש הגרפיקה |
| os | מחלקה שעובדת עם מערכת ההפעלה |
| hashlib | מחלקה כדי לבצע גיבוב לסיסמאות |
| pydub | מחלקת המוזיקה כדי לבצע שינויים בקובץ מוזיקה |
| random | כדי ליצור מפתח רנדומלי להעברה בדיפי הלמן |
| pubsub | להעברת הודעות בין הgui ללוגיקה |
| time | כדי ליצור בלוקינג |
| base64 | בשביל הצפנה |
| Cryptodome | בשביל הצפנה |

## מודולים שלי:

### צד לקוח:

| **ClientCom** | | | |
| --- | --- | --- | --- |
| מחלקת תקשורת לקוח עם השרת | | | |
| **משתנים** | | | |
| server\_ip | | כתובת הip של השרת | |
| port | | port התקשורת | |
| my\_socket | | Socket התקשורת של הלקוח | |
| encrypt | | עצם הצפנה | |
| msg\_q | | תור לקבלת ההודעות מהשרת | |
| running | | האם התקשורת עדיין פעילה | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_(server\_ip,port,msg\_q,encrypt) | מקבלת ip , port , msg\_q ליצירת עצם תקשורת ומקבלת עצם להצפנה | | יוצרת עצם תקשורת וקוראת לthread main\_loop() |
| main\_loop()\_ | לא מקבלת פרמטרים | | יוצרת תקשורת עם השרת , יוצרת socket  קוראת לפעולה \_difi()  ומתחילה לרוץ על הלולאה הראשית בה היא קולטת הודעות מהשרת ומכניסה אותן לתור |
| send(msg) | מקבלת הודעה לשליחה | | שולחת את אורך ההודעה ואת ההודעה **מוצפנת** לשרת בשתי שליחות  שליחת אורך ב5 בתים  ושליחת ההודעה |
| \_diffie() | לא מקבלת כלום | | עושה תהליך חישוב מפתח סימטרי להצפנה(דיפי הלמן) עם השרת ומחזירה את המפתח |
| send\_file(file\_path) | מקבלת כתובת של הקובץ | | פותחת את הקובץ, שולחת את הקובץ בשתי שליחות:  שליחת אורך הקובץ 8 בתים  שליחת הקובץ עצמו |
| recv\_file(file\_name ,start\_time, volume, proto) | מקבלת אורך קובץ שם קובץ זמן התחלה, ווליום, ופרוטוקול | | מקבלת את הקובץ אותו ומכניסה לתור |

| **ClientLogic** | | | |
| --- | --- | --- | --- |
| מחלקת main לוגיקת לקוח, מטפלת בקלט לקוח והודעות מהשרת | | | |
| **משתנים** | | | |
| strips | | מילון לשמירת הרצועות של הקובץ הנוכחי שהלקוח עובד עליו המילון מכיל שם רצועה ואת עצם רצועת המוזיקה | |
| msg\_q | | תור להודעות מהשרת הראשי | |
| file\_q | | תור לקבלת קבצים משרת הקבצים | |
| cp | | עצם הצפנה | |
| comm | | עצם לביצוע תקשורת עם השרת | |
| file\_comm | | עצם לביצוע תקשורת עם שרת קבצים | |
| strips\_count | | מונה בשביל לדעת כמה רצועות צריך לקלוט | |
| play | | עצם להשמעת מוזיקה | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| main() | לא מקבלת כלום | | יוצרת עצם ClientLogic, יוצרת את כל הpubsubs, מתחילה את לולאת הגרפיקה, מוחקת את כל הקבצים מclient\_music בסוף ריצה |
| \_\_init\_\_() | לא מקבלת כלום | | מאתחלת את כל הערכים של המחלקה, יוצרת עצם ClientCom וקוראת לתהליכון handle\_msgs() |
| handle\_msgs() | לא מקבלת כלום | | לולאה אין סופית שמוציאה הודעות מmsg\_q ומטפלת בהן |
| handle\_files() | לא מקבלת כלום | | לולאה אין סופית שמוציאה הודעות מfile\_q ומטפלת בהן |
| add\_strip(strip\_name, starting\_point, volume, file\_path) | מקבלת שם רצועה, זמן התחלה, ווליום, וכתובת קובץ | | יוצרת עצם MusicStrip ומוסיפה אותו לself.strips, קוראת לפעולה שמעדכנת שינויים בממשק משתמש |
| layer\_music() | לא מקבלת כלום | | יוצרת קובץ מוזיקה להשמעה שמורכב מכל הרצועות של הלקוח |
| play\_music(song, song\_time) | מקבלת קובץ שיר להשמעה וזמן שיר | | תהליכון שמשמיעה את השיר |
| stop\_play\_music() | לא מקבלת כלום | | מפסיקה להשמיע את המוזיקה, דואגת שהממשק משתמש יתעדכן |
| mute\_unmute(strip\_name) | מקבלת שם רצועה | | משתיקה או לא משתיקה אותה |
| inc(strip\_name) | מקבלת שם רצועה | | מגבירה ווליום של רצועה |
| dec(strip\_name) | מקבלת שם רצועה | | מנמיכה ווליום של רצועה |
| cut\_beginning(strip\_name, time) | מקבלת שם רצועה וזמן לחתוך | | מורידה את time מההתחלה של הרצועה |
| cut\_end(strip\_name, time) | מקבלת שם רצועה וזמן לחתוך | | מורידה את time מהסוף של הרצועה |
| graphic\_loop() | לא מקבלת כלום | | מפעילה את הגרפיקה של ממשק המשתמש |

| **Encrypt** | | | |
| --- | --- | --- | --- |
| מחלקה לביצוע הצפנה ופיענוח סימטרי | | | |
| **משתנים** | | | |
| encrypt\_keys | | מפתחות הצפנה | |
| bs | | block size | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_() | לא מקבלת כלום | | יוצרת עצם להצפנה |
| add\_key(key,ip=None) | מקבלת מפתח ויכולה לקבל גם ip של לקוח | | מוסיפה את המפתח ל encrypt\_keys |
| encrypt(raw,ip =None) | מקבלת הודעה להצפין ויכולה לקבל גם ip של לקוח | | מחזירה הודעה מוצפנת עם המפתח ע"פ המילון encrypt\_keys |
| decrypt(raw,ip =None) | מקבלת הודעה לפיענוח ויכולה לקבל גם ip של לקוח | | מחזירה הודעה מפוענחת עם המפתח ע"פ המילון encrypt\_keys |
| \_pad(s) | מקבלת בתים כדי לעשות להם padding | | מחזירה בתים שעשו להם padding |
| hash(message) | מקבלת הודעה | | מחזירה גיבוב של ההודעה |

| **MusicStrip** | | | |
| --- | --- | --- | --- |
| מחלקה לבניית עצם רצועת מוזיקה | | | |
| **משתנים** | | | |
| file | | קובץ מוזיקה | |
| length | | האורך של הרצועה | |
| name | | שם הקובץ | |
| muted | | אומר אם הרצועה מושתקת או לא | |
| volume | | ווליום הרצועה | |
| starting\_point | | מתי הרצועה מתחילה להתנגן | |
| path | | כתובת כדי לשמור את הקובץ | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_(file,name,starting\_point,volume) | מקבלת קובץ בבתים וזמן התחלה ,ווליום | | יוצרת עצם רצועת מוזיקה ושומרת את הקובץ על המחשב |
| trim\_start(time) | מקבלת זמן להסיר | | מקצרת את את הרצועה מההתחלה לפי time מחזירה אם הקובץ קוצר |
| trim\_end(time) | מקבלת זמן להסיר | | מקצרת את את הרצועה מהסוף לפי time מחזירה אם הקובץ קוצר |
| mute\_unmute() | לא מקבלת כלום | | אם mute הוא true אז היא משנה אותו לfalse ולהפך |
| set\_strip(file,starting,volume) | מקבלת קובץ בבתים ,זמן התחלה, ווליום קובץ | | מעדכנת את כל הנתונים של ברצועה ושומרת אותה מחדש |
| save\_file() | לא מקבלת כלום | | שומרת את הרצועת מוזיקה |

| **ClientProtocol** | | |
| --- | --- | --- |
| מחלקה לבניית הודעות לשליחה לפי פרוטוקול הלקוח | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| log\_in(username,password) | מקבלת שם משתמש וסיסמא | מחזירה הודעה שבנויה לפי הפרוטוקול |
| sign\_in(username\_password) | מקבלת שם משתמש וסיסמא | מחזירה הודעה שבנויה לפי הפרוטוקול |
| create\_file(file\_name) | מקבלת שם קובץ עריכה משותף | מחזירה הודעה שבנויה לפי הפרוטוקול |
| start\_edit\_file(file\_name) | מקבלת שם קובץ עריכה משותף | מחזירה הודעה שבנויה לפי הפרוטוקול |
| shorten\_start(strip\_name, time\_to\_take) | מקבלת שם רצועה וזמן להוריד | מחזירה הודעה שבנויה לפי הפרוטוקול |
| shorten\_end(strip\_name, time\_to\_take) | מקבלת שם רצועה וזמן להוריד | מחזירה הודעה שבנויה לפי הפרוטוקול |
| move\_strip(strip\_name,start\_time) | מקבלת שם רצועה וזמן התחלה של רצועה על ציר הזמן | מחזירה הודעה שבנויה לפי הפרוטוקול |
| inc\_volume(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| dec\_volume(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| undo(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| start\_edit\_strip(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| delete\_strip(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| strip\_name\_request (strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| add\_strip(strip\_name) | מקבלת שם רצועה זמן | מחזירה הודעה שבנויה לפי הפרוטוקול |
| send\_start\_time(start\_time) | מקבלת זמן התחלה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| stop\_editing\_strip(strip\_name) | מקבלת שם רצועה זמן | מחזירה הודעה שבנויה לפי הפרוטוקול |
| send\_volume(volume) | ווליום | מחזירה הודעה שבנויה לפי הפרוטוקול |
| unpack(msg) | הודעה מורכבת לפי הפרוטוקול | מפרקת ומחזירה opcode, list of params |

| **Program** | | | |
| --- | --- | --- | --- |
| מחלקת גרפיקה מסגרת שמכילה בתוכה את כל הפאנלים | | | |
| **משתנים** | | | |
| comm | | עצם תקשורת של הלקוח | |
| file\_comm | | עצם תקשורת של לקוח הקבצים | |
| username | | שם המשתמש של הלקוח | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_(comm) | מקבלת עצם תקשורת מהשרת | | מאתחלת את כל העצמים של הפאנלים ומציגה את ממשק המשתמש |

| **SignIn** | | |
| --- | --- | --- |
| מחלקת מסך הרשמה | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| \_\_init\_\_(frame) | מקבלת מסגרת של המחלקה הראשית | יוצרת עצם מסך הרשמה |

| **LogIn** | | |
| --- | --- | --- |
| מחלקת מסך התחברות | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| \_\_init\_\_( frame) | מקבלת מסגרת של המחלקה הראשית | יוצרת עצם מסך התחברות |

| **Home** | | |
| --- | --- | --- |
| מחלקת מסך בית | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| \_\_init\_\_( frame) | מקבלת מסגרת של המחלקה הראשית | יוצרת עצם מסך בית |

| **AddFile** | | |
| --- | --- | --- |
| מחלקת מסך הוספת קובץ | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| \_\_init\_\_( frame) | מקבלת מסגרת של המחלקה הראשית | יוצרת עצם מסך הוספת קובץ |

| **EditFile** | | |
| --- | --- | --- |
| מחלקת מסך עריכת קובץ | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| \_\_init\_\_( frame) | מקבלת מסגרת של המחלקה הראשית | יוצרת עצם מסך עריכת קובץ |

| **AddStrip** | | |
| --- | --- | --- |
| מחלקת מסך הוספת רצועה | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| \_\_init\_\_( frame) | מקבלת מסגרת של המחלקה הראשית | יוצרת עצם מסך הוספת רצועה |

## צד שרת

|  |  |  |  |
| --- | --- | --- | --- |
| **ServerCom** | | | |
| מחלקת תקשורת שרת עם הלקוח | | | |
| **משתנים** | | | |
| port | | port התקשורת | |
| my\_socket | | socket לביצוע התקשורת | |
| msg\_q | | תור לקבלת ההודעות מהלקוח | |
| open\_clients | | מילון שמכיל מפתח socket וערך ip | |
| encrypt | | עצם Encrypt לביצוע הצפנות לפני שליחת הודעות | |
| running | | אם התקשורת פעילה | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_(port,msg\_q,encrypt) | מקבלת port לתקשרות ותור לקבלת הודעות, מקבלת עצם לביצוע הצפנה | | יוצרת עצם תקשרות שרת וקוראת לthread של \_main\_loop() |
| \_main\_loop() | לא מקבלת כלום | | יוצרת את socket השרת, יוצרת תקשורת עם הלקוחות ולכל ip חדש היא קוראת ל \_diffie() ,מכניסה את כל ההודעות המתקבלות לתור ההודעות |
| \_diffie(client,addr) | מקבלת סוקט וכתובת IP של לקוח | | עושה תהליך חישוב מפתח סימטרי להצפנה(דיפי הלמן) ומחזירה את המפתח |
| send(ips,msg) | מקבלת רשימה של ips לשלוח להם הודעה | | שולחת את ההודעה **מוצפנת** לכל הסוקטים שנמצאים ips בשתי שליחות:  שליחת אורך ההודעה ב 5 בתים  שליחת ההודעה עצמה |
| \_disconnect\_client(socket) | מקבלת סוקט לנתק | | מנתקת את הסוקט ומוחקת אותו מ open\_clients שמה הודעה בmsg\_q שהשרת יסיר את הלקוח |
| find\_socket\_by\_ip(Ip) | מקבלת ip | | מחזירה Socket |
| send\_file(socket,file\_path) | מקבלת socket ו כתובת של הקובץ | | שולחת ללקוח הנדרש את הקובץ **מוצפן** בשתי שליחות :  אורך הקובץ ב 8בתים  ,הקובץ עצמו |
| recv\_file(file\_name,start\_time,volume, socket,proto) | מקבלת שם קובץ זמן התחלה, ווליום, וסוקט לקבל ממנו מידע, ופרוטוקול | | קולטת את הקובץ מהלקוח מחזירה את הקובץ בבתים עם הפרטים שלו |

| **ServerLogic** | | | |
| --- | --- | --- | --- |
| מחלקת main לוגיקת שרת, מטפלת בהודעות מהלקוח | | | |
| **משתנים** | | | |
| usernames | | מילון שמכיל מפתח ip וערך שם משתמש | |
| msg\_q | | תור ההודעות מהלקוח | |
| editing\_groups | | מילון שמורכב ממפתח שם קובץ עריכה משותף ומערך עצם EditGroup | |
| cp | | עצם הצפנה | |
| comm | | עצם לביצוע תקשורת | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_() | לא מקבלת כלום | | יוצרת את כל הערכים של המחלקה ומפעילה את התהליכון handle\_msgs |
| main() | לא מקבלת כלום | | יוצרת עצם ServerLogic |
| handle\_msgs() | לא מקבלת כלום | | תהליכון של לולואה שמוציא הודעות מהתור ומטפל בהן |
| log\_in(ip, password, username) | מקבלת ip, סיסמא ושם משתמש | | בודקת אם הלקוח לא מחובר כבר למערכת, בודקת עם בסיס הנתונים אם הסיסמא תקינה, אם כן מוסיפה את הלקוח לusernames שולחת לו שהסיסמא נכונה, שולחת ללקוח את קבצי העריכה שלו, ושולחת ללקוח את כל שמות המשתמשים של המערכת, אם הסיסמא ששגויה שולחת ללקוח שהסיסמא שגויה |
| sign in(ip, password, username) | מקבלת ip, סיסמא ושם משתמש | | בודקת עם בסיס הנתונים אם המשתמש לא קיים כבר, אם המשתמש לא קיים מוסיפה את הלקוח לusernames שולחת לו שהמשתמש התווסף, שולחת ללקוח את קבצי העריכה שלו, ושולחת ללקוח את כל שמות המשתמשים של המערכת, מוסיפה את המשתמש לבסיס הנתונים וקוראת לsend\_add\_user, אם המשתמש כבר קיים במערכת שולחת ללקוח שהמשתמש קיים כבר |
| edit\_file(ip, file\_name) | מקבלת ip של לקוח ושם של קובץ עריכה משותף | | אם לא קיים לקבוצה עצם עריכה היא יוצרת לקבוצה עצם EditGroup ומוסיפה אותו לediting\_groups, מוחקת את הלקוח מcomm.open\_clients |
| create\_edit\_file(ip, file\_name) | מקבלת ip ושם קובץ עריכה משותף | | בודקת עם בסיס הנתונים שהקובץ לא קיים כבר, אם הקובץ קיים היא שולחת ללקוח שהקובץ קיים כבר במערכת, אם הקובץ לא קיים היא יוצרת את הקובץ בבסיס הנתונים, יוצרת תיקיה בשביל הקובץ, קוראת לפעולה send\_add\_file() ולפעולה edit\_file() |
| send\_add\_user(username, ip) | מקבלת שם משתמש וip | | שולחת לכל הלקוחות המחוברים על משתמש שהתווסף למערכת |
| send\_add\_file(file\_name, ip) | מקבלץ שם קובץ עריכה משותף וip | | שולחת לכל הלקוחות המחוברים על משתמש שהתווסף למערכת |

| **EditGroup** | | | |
| --- | --- | --- | --- |
| מחלקה לביצוע עבודת עריכת מוזיקה בקבוצה | | | |
| **משתנים** | | | |
| runing | | משתנה בוליאני אם המשתמשים עדיין עורכים | |
| msg\_q | | תור הודעות מהלקוח | |
| server\_q | | תור ההודעות של השרת הראשי, כדי להעביר הודעות לשרת הראשי | |
| members | | רשימה של כל שמות חברי הקבוצה | |
| file\_name | | שם הקובץ | |
| strips | | מילון עם מפתח שם רצועה וערך עצם MusicStrip | |
| changes | | מילון שמורכב ממפתח שם רצועה וערך שהוא רשימה של כל המצבים של הרצועה לפני השינוי האחרון | |
| file\_servers | | מילון עם מפתח ip וערך עצם ServerComm | |
| comm | | עצם תקשורת | |
| file\_port | | פורט של שרתי קבצים | |
| encrypt | | עצם הצפנה | |
| group\_name | | שם קובץ עריכה משותף | |
| file\_queues | | מילון עם מפתח ip וערך תור הודעות | |
| comm | | עצם EditGroupCom לביצוע תקשורת עם השרת | |
| usernames | | מילון עם מפתח ip וערך שם משתמש | |
| strip\_edit | | מילון של כל המשתמשים שעורכים רצועה כלשהי, מפתח שם רצועה, שם משתמש | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_(socket,username, group\_name, server\_q, encrypt, ip) | מקבלת סוקט, שם משתמש, שם קובץ עריכה משותף, תור הודעות של השרת הראשי, עצם הצפנה, ip של לקוח | | מאתחלת את כל הערכים של המחלקה, קוראת לפעולה set\_strips() יוצרת ללקוח שהתחבר ServerCom לקבצים ומוסיפה אותו לfile\_servers שולחת ללקוח הודעה לפתוח לקוח קבצים, קוראת לפעולה send\_edit\_file() קוראת לתהליכון handle\_msgs() ולתהליכון handle\_files |
| handle\_msgs() | לא מקבלת כלום | | תהליכון לולאה שמוציאה הודעות מהתור ומטפלת בהן |
| handle\_files(file\_q) | מקבלת תור של קבצים | | תהליכון לולאה שמוציאה את הקבצים מהתור לכל קובץ קוראת לפעולה add\_strip |
| add\_editor(socket, username, ip) | מקבלת סוקט ושם משתמש וip | | מוסיפה את הלקוח לcomm.open\_clients מוסיפה את הלקוח לusernames מוסיפה ללקוח תור הודעות קבצים ומוסיפה אותו לfile\_queues יוצרת ללקוח עצם ServerCom לקבצים ומוסיפה אותו לfile\_servers שולחת ללקוח הודעה לפתוח שרת קבצים, קוראת לפעולה send\_edit\_file() |
| send\_edit\_flie(ip) | מקבלת ip | | שולחת הודעה ללקוח עם כל הנתונים של קובץ העריכה המשותף לפי הפרוטוקול |
| inc(ip, strip\_name) | מקבלת ip ושם רצועה | | מגבירה את הווליום לרצועה ושולחת הודעה לשאר הלקוחות להגביר את הווליום לרצועה |
| dec(ip, strip\_name) | מקבלת ip ושם רצועה | | מנמיכה את הווליום לרצועה ושולחת הודעה לשאר הלקוחות להנמיך ווליום לרצועה |
| add\_strip(strip\_name, start\_time, volume, file, ip) | מקבלת שם רצועה, זמן התחלה, ווליום, קובץ בבתים, וip | | שומרת את הקובץ, יוצרת עצם MusicStrip ומוסיפה אותו לstrips , שולחת לשארא הלקוחות הודעה להוסיף את הרצועה |
| send\_edit\_file(ip) | מקבלת ip | | שולחת את כל הרצועות של קובץ העריכה המשותף לip |
| start\_edit\_strip(ip, strip\_name) | מקבלת ip ושם רצועה | | בודקת אם אף אחד לא עורך הרצועה strip\_name ושולחת לפי זה הודעה ללקוח אם הוא יכול או לא לערוך את הרצועה, אם הלקוח יכול לערוך את הרצועה שולחת לשאר הלקוחות שהמשתמש הזה עורך את הרצועה ומוסיפה את הלקוח לstrip\_edit |
| stop\_editing\_strip(ip) | מקבלת ip | | מוחקת את הלקוח מstrip\_edit ושולחת הודעה לכל שאר הלקוחות שהמשתמש הזה הפסיק לערוך |
| remove\_client\_editing(ip) | מקבלת ip | | מוצאת את איזו רצועה המשתמש עורך וקוראת לפעולה stop\_editing\_strip() |
| cut\_beginning(ip, strip\_name, cut\_tim) | מקבלת ip , שם רצועה, וזמן לחתוך | | חותכת מההתחלה של הרצועה את cut\_time ושולחת לשאר הלקוחות לחתוך את הרצועה מההתחלה |
| cut\_end(ip, strip\_name, cut\_time) | מקבלת ip , שם רצועה, וזמן לחתוך | | חותכת מהסוף של הרצועה את cut\_time ושולחת לשאר הלקוחות לחתוך את הרצועה מהסוף |
| check\_strip\_exists(ip, strip\_name) | מקבלת ip ושם רצועה | | בודרת אם הרצועה קיימת כבר במערכת ושולחת הודעה לip אם היא קיימת או לא |
| set\_strips() | לא מקבלת כלום | | מאתחלת את strips לפי המידע ששמור בבסיס הנתונים |
| save\_file() | לא מקבלת כלום | | שומרת את כל הקבצים של הרצועות למחשב השרת ומוחקת את כל הקבצים מתיקיית undo |

| **EditGroupCom** | | | |
| --- | --- | --- | --- |
| מחלקה לביצוע תקשורת בין חברי קבוצה שעורכים מוזיקה | | | |
| **משתנים** | | | |
| msg\_q | | תור הודעות מחברי הקבוצה | |
| open\_clients | | מילון שמורכב ממפתח סוקט וערך שם משתמש | |
| running | | משתנה בוליאני ללולאה הראשית אם התקשורת עדיין פעילה | |
| encrypt | | עצם Encrypt לביצוע הצפנות לפני שליחת הודעות | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_(socket, ip,msg\_q,encrypt) | מקבלת סוקט של לקוח ,שם משתמש של לקוח ,תור הודעות ועצם Encrypt לביצוע הצפנה | | מוסיפה את הלקוח לopen\_clients וקוראת לתהליכון הראשי main\_loop() |
| main\_loop() | לא מקבלת כלום | | מתחילה לולאה ראשית שמקבלת הודעות מהלקוחות ומכניסה אותן לmsg\_q |
| disconnect\_client(socket) | מקבלת סוקט לסגור | | סוגרת את התקשורת עם הלקוח, מוחקת אותו מopen\_clients ומכניסה הודעת לmsg\_q על הלקוח שנסגר |
| send(ips,msg) | מקבלת רשימה של ips לשלוח להם הודעה | | שולחת את ההודעה **מוצפנת** לכל הסוקטים שנמצאים בsockets |
| \_disconnect\_client(socket) | מקבלת סוקט | | סוגרת את הסוקט ומוחקת אותו מopen\_clients |
| \_find\_socket\_by\_ip(ip) | מקבלת ip | | מחזירה socket לאותו הip |

| **ServerProtocol** | | |
| --- | --- | --- |
| מחלקה לבניית הודעות לשליחה לפי פרוטוקול השרת | | |
| **פעולות** | | |
| **חתימת הפעולה** | **טענת כניסה** | **טענת יציאה** |
| usernames(usernames) | מקבלת רשימה של כל המשתמשים הקיימים במערכת | מחזירה הודעה שבנויה לפי הפרוטוקול |
| edit\_files(edit\_files) | מקבלת רשימה של שמות קבצי עריכה משותפים | מחזירה הודעה שבנויה לפי הפרוטוקול |
| added\_file(file\_name) | מקבלת שם של קובץ עריכה משותף | מחזירה הודעה שבנויה לפי הפרוטוקול |
| added\_user(username) | מקבלת שם משתמש | מחזירה הודעה שבנויה לפי הפרוטוקול |
| shorten\_start(strip\_name, time\_to\_take) | מקבלת שם רצועה וזמן להוריד | מחזירה הודעה שבנויה לפי הפרוטוקול |
| shorten\_end(strip\_name, time\_to\_take) | מקבלת שם רצועה וזמן להוריד | מחזירה הודעה שבנויה לפי הפרוטוקול |
| move\_strip(strip\_name,start\_time) | מקבלת שם רצועה וזמן התחלה של רצועה על ציר הזמן | מחזירה הודעה שבנויה לפי הפרוטוקול |
| inc\_volume(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| dec\_volume(strip\_name) | מקבלת שם רצועה | מחזירה הודעה שבנויה לפי הפרוטוקול |
| undo(strip\_name) | מקבלת שם רצועה | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| editing\_members(usernames) | מקבלת רשימה של שמות | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| start\_edit\_strip(strip\_name,username) | מקבלת שם רצועה ושם משתמש | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| stop\_edit\_strip(strip\_name) | מקבלת שם רצועה | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| delete\_strip(strip\_name) | מקבלת שם רצועה | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| file\_server\_port(port) | מקבלת port | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| strip\_name(strip\_name) | מקבלת שם רצועה | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| send\_start\_time(start\_time) | מקבלת זמן התחלת רצועה על ציר הזמן | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| send\_volume(volume) | מקבלת ווליום | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| strips\_amount(amount) | מקבלת כמות רצועות | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| strip\_occupied(editor\_name) | מקבלת שם משתמש | מחזירה את ההודעה שבנויה לפי הפרוטוקול |
| unpack(msg) | הודעה מורכבת לפי הפרוטוקול | מפרקת ומחזירה opcode, list of params |

| **DB** | | | |
| --- | --- | --- | --- |
| מחלקה לבניית עצם DB לקריאה ושינוי של נתונים במסדי נתונים | | | |
| **משתנים** | | | |
| conn | | Sqlite3.Connection | |
| cursor | | Sqlite.Cursor | |
| db\_name | | שם בסיס הנתונים | |
| users\_table | | שם טבלת המשתמשים | |
| editors\_table | | שם הטבלה העורכים | |
| **פעולות** | | | |
| **חתימת הפעולה** | **טענת כניסה** | | **טענת יציאה** |
| \_\_init\_\_() | לא מקבלת כלום | | יוצרת עצם DB וקוראת לפעולה \_create\_db() |
| \_create\_db() | לא מקבלת כלום | | מתחבר למסד הנתונים ויוצרת את הטבלאות אם הן לא קיימות |
| \_username\_exists(username) | מקבלת שם משתמש | | מחזירה True אם המשתמש קיים וFalse אם הוא לא קיים |
| add\_user(username,password) | מקבלת שם משתמש וסיסמא מגובבת | | מקבלת שם משתמש וסיסמא מגובבת מחזירה True אם היא הוסיפה את המשתמש וFalse אם שם המשתמש תפוס |
| check\_password(username,password) | מקבלת שם משתמש וסיסמא מגובבת | | מקבלת שם משתמש וסיסמא מגובבת מחזירה True אם הסיסמא תואמת לשם המשתמש וFalse אם היא לא תואמת |
| \_file\_exists(file\_name,group\_members) | מקבלת שם של קובץ עריכה משותף ומחרוזת של שמות חברי הקבוצה | | בודקת אם לקבוצה יש כבר קובץ באותו השם מחזירה True אם הוא קיים וFalse אם הוא לא קיים |
| add\_file(file\_name,group\_members) | מקבלת שם של קובץ עריכה משותף ומחרוזת של שמות חברי הקבוצה | | מוסיפה את הקובץ אם אין לקבוצה קובץ קיים באותו השם מחזירה True אם הקובץ נוסף וFalse אם הוא לא נוסף |
| update\_file(file\_name\_group\_members,details) | מקבלת שם של קובץ עריכה משותף ומחרוזת של שמות חברי הקבוצה ונתונים לשינוי בקובץ | | מעדכנת את הקובץ מוחקת את הנתונים הקודמים שלו בעמודת strips ושמה את details במקום |
| get\_details(file\_name,group\_members) | מקבלת שם של קובץ עריכה משותף ומחרוזת של שמות חברי הקבוצה | | מחזירה את הנתונים של הקובץ |
| get\_users(username) | מקבלת שם משתמש | | מחזירה את כל השמות של המשתמשים הקיימים במערכת שהם לא username |
| get\_user\_files(username) | מקבלת שם משתמש | | מחזירה את כל הקבצים המשותפים של המשתמש |
| close\_db | לא מקבלת כלום | | סוגרת את בסיס הנתונים |

## תיאור סביבת הפיתוח

שפת התכנות בה בחרתי לעבוד בה לכתיבת הפרויקט היא python 3.7 כדי לכתוב את קוד הלקוח קוד השרת ומימוש התקשורת בsocket – ים. בנוסף אני משתמש גם בשפת תכנות SQL בשביל מימוש ותחזוקה של בסיסי נתונים.

### כלי פיתוח:

Pycharm- לכתיבת הקוד ופיתוח המערכת

### כלי בדיקה:

Wireshark - לבדיקת התקשורת

PyCharm – שימוש בדיבאגר לניפוי שגיאות

Db\_Browser – לבדיקת בסיסי הנתונים

fileExplorer – לראות שאכן נוצרים קבצים

media\_player – לשמיעת המוסיקה

## תיאור האלגוריתמים המרכזיים בפרויקט

### בעיה 1. איך תתבצע התקשורת?

|  |  |  |
| --- | --- | --- |
| **אפשרויות** | **יתרונות** | **חסרונות** |
| יצירת socket אחד אצל השרת והלקוח שיטפל בכל התקשורת(קבצים וטקסט) | יחסית פשוט וקל למימוש. | יכול להיווצר עומס בשרת בגלל שכל המידע עובר דרך socket אחד |
| יצירת 2 socketים אצל השרת והלקוח, אחד לתקשורת הרגילה ואחד להעברת קבצים | מפריד בין 2 סוגי תקשורת שונים ומקל על העומס בצד השרת | יוצר עומס אצל צד השרת בגלל שעדיין כל הקבצים של כל הלקוחות עוברים דרך socket יחיד |
| יצירת socket אחד לתקשורת אצל השרת ואצל הלקוח, יצירת socket להעברת קבצים אצל הלקוח ועבור כל socket כזה לפתוח socket חדש אצל השרת כאשר כל תקשורת כזאת יושבת על port אחר כדי למנוע עומס | מפריד בין תקשורת רגילה לתקשורת של קבצים ומונעת עומס כי זה מפזר את התקשורת על מספר PORTים | מסובך יותר למימוש מצריך יצירה רבה של SOCKETים |

בסוף בחרתי באפשרות השלישית בגלל שהיא מאפשרת לי להעביר מידע בצורה הכי יעילה וגם היא יוצרת הכי פחות עומס על המערכת.

### בעיה 2. איך תתבצע העבודה במקביל?

|  |  |  |
| --- | --- | --- |
| **אפשרויות** | **יתרונות** | **חסרונות** |
| לכל משתמש תהיה רצועת מוזיקה משלו בקובץ העריכה המשותף והוא יוכל לערוך אך ורק אותה ולא את הרצועות האחרות | דואגת שהמשתמשים לא ידרסו את העבודה אחד של השני תוך כדי עריכה ובכך לאפשר תוצר סופי יותר טוב | מגבילה את המשתמשים לעבודה רק על רצועה אחת |
| כל משתמש יוכל לעבוד על איזה רצועה שהוא רוצה בהנחה שהיא לא תפוסה(שמשתמש אחר לא עורך אותה) וברגע שהרצועה תתפנה הוא יוכל לערוך אותה. | דואגת שהמשתמשים לא ידרסו את העבודה אחד של השני תוך כדי עריכה וגם לא מגבילה את המשתמש לרצועה אחת שהוא יכול לערוך | מאפשרת למשתמש להרוס עריכה שמשתמש אחר עשה |

בסוף בחרתי באפשרות השנייה בגלל שהיא מאפשרת לכל המשתמשים לתת את הנגיעה היצירתית שלהם בכל אחד מהרצועות וגם היא מאפשרת עבודה מסודרת במקביל ללא דריסות

### בעיה 3. מה תהיה שיטת ההצפנה?

|  |  |  |
| --- | --- | --- |
| **אפשרויות** | **יתרונות** | **חסרונות** |
| הצפנה א סימטרית | מאוד בטוחה ונורא קשה לפיענוח בגלל שיש כמה מפתחות | יחסית כבדה ולוקחת הרבה מאוד זמן לפיענוח, פחות מהירה |
| הצפנה סימטרית והעברת מפתח בתקשורת | מהירה ופשוטה יותר למימוש | פחות בטוחה ומעבירה את המפתח גלוי ברשת מה שפותח את המערכת לסיכונים |
| הצפנה סימטרית והעברת מפתח ברשת עם פרוטוקול דיפי הלמן | מהירה יותר מהצפנה סימטרית ומאפשרת לדאוג שהמפתח לא יהיה גלוי ברשת בגלל שהיא מעבירה אותו בעזרת פרוטוקול דיפי הלמן | פחות בטוחה מהצפנה סימטרית |

בסוף בחרתי באפשרות השלישית מכיוון שהיא מאפשרת לי להעביר מידע בצורה הכי מהירה וגם מבלי לגלות את מפתח ההצפנה ברשת.

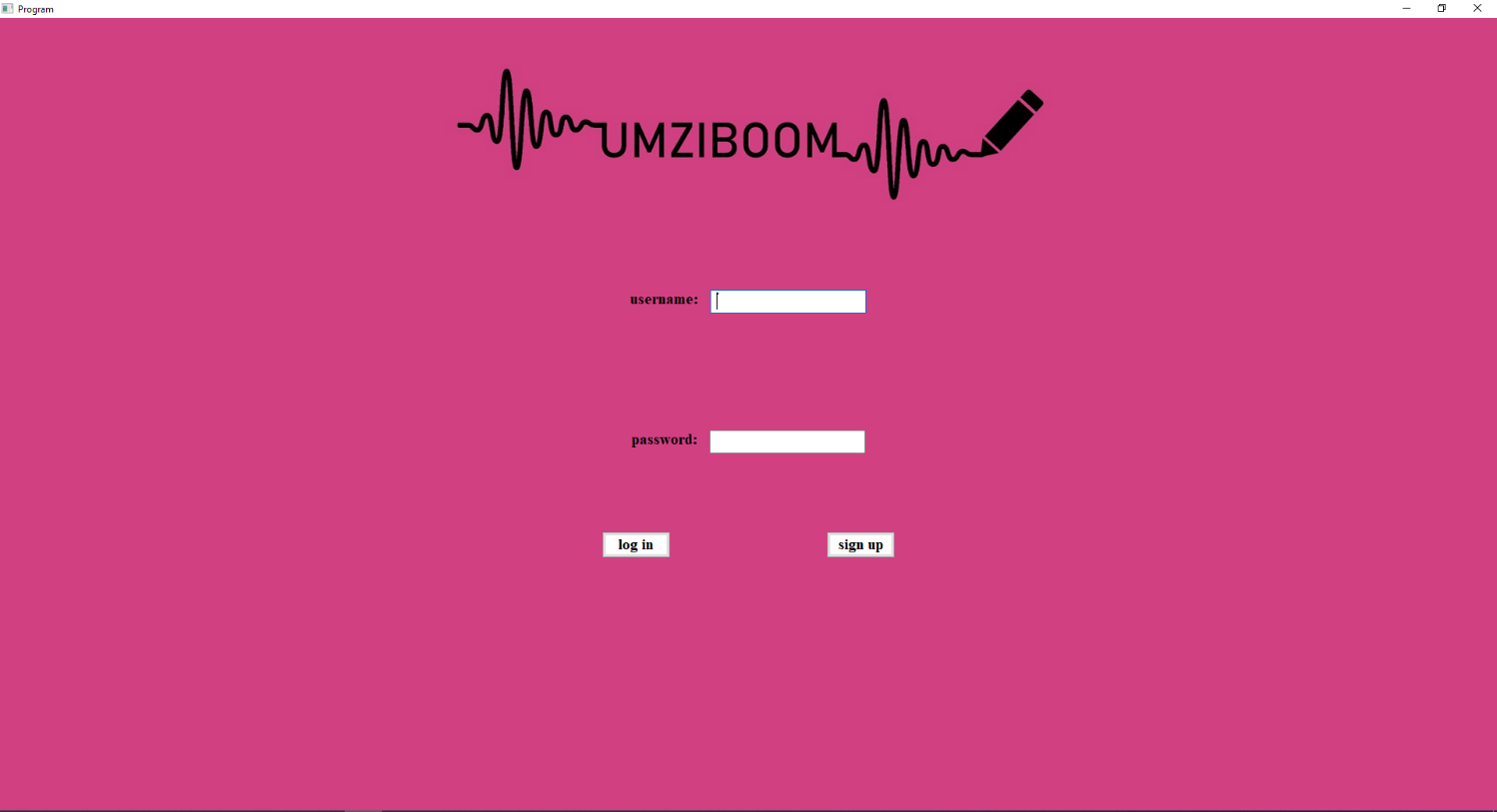
### בעיה 4. איך תתבצע השמעת המוזיקה?

|  |  |  |
| --- | --- | --- |
| **אפשרויות** | **יתרונות** | **חסרונות** |
| שמירת קבצי המוזיקה אך ורק אצל השרת וברגע שהלקוח ירצה לשמוע את המוזיקה הוא ישלח בקשה לשרת והוא ישלח לו את הקבצים ואז ההשמעה תתבצע בצורת streaming | חסכונית במקום בגלל שהיא מצריכה שמירת קבצים אך ורק אצל השרת | מסובכת למימוש ויכולה להיות בעייתית אם יש בעיות באינטרנט |
| שמירת קבצי המוזיקה אצל השרת ואצל הלקוח ואז אם הלקוח ירצה לשמוע מוזיקה הוא ייגש לקבצים ששמורים אצלו במחשב וכשהוא יסיים לערוך הוא ימחק את הקבצים מהמחשב שלו | מאפשרת השמעה יותר מהירה ובטוחה של המוזיקה כי הקבצים שמורים כבר אצל הלקוח | פחות חסכונית במקום בגלל שהיא מצריכה לשמור את קבצי המוזיקה גם אצל הלקוח וגם אצל השרת |

בסוף בחרתי באפשרות השנייה בגלל שהיא מאפשרת לי לגשת למוזיקה בצורה הכי מהירה וגם היא לא מעמיסה על הזיכרון של המחשב בגלל שהקבצים נשמרים על המחשב זמנית.

## תיאור מסכי הפרויקט:

### מסך כניסה התחברות



### מסך יצירת משתמש

### מסך בית

תמונה שמכילה טקסט, צילום מסך, גופן, גרפיקה

התיאור נוצר באופן אוטומטי

### תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב התיאור נוצר באופן אוטומטימסך יצירת קובץ

### מסך עריכת קובץ

תמונה שמכילה טקסט, צילום מסך, תכונות מולטימדיה, תוכנה גרפית

התיאור נוצר באופן אוטומטי

### תמונה שמכילה טקסט, צילום מסך, תכונות מולטימדיה, תוכנה התיאור נוצר באופן אוטומטיעריכת רצועה

### תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה התיאור נוצר באופן אוטומטיהוספת רצועה

## תיאור פרוטוקול התקשורת

### צד לקוח:

|  |  |  |  |
| --- | --- | --- | --- |
| שם הפרוטוקול | אורך הודעה 5 ביטים | מספר בקשה 2 ביטים | מבנה ההודעה |
| בקשת התחברות לשרת | אורך הודעה | 1 | סיסמא|שם משתמש |
| בקשת הרשמה לשרת | אורך הודעה | 2 | סיסמא|שם משתמש |
| בקשת יצירת קובץ עריכה משותף | אורך הודעה | 3 | שם קובץ;חבר1,חבר2,חבר3,חבר4 |
| בקשת התחלת עריכה בקובץ עריכה משותף קיים | אורך הודעה | 4 | שם קובץ;חבר1,חבר2,חבר3,חבר4 |
| בקשת הפסקת עריכה של קובץ עריכה משותף | אורך הודעה | 5 | אין הודעה |
| שליחת הודעה בצ'אט | אורך הודעה | 6 | הודעת טקסט |
| שליחת הודעה על קיצור רצועה מההתחלה | אורך הודעה | 7 | שם רצועה|זמן להוריד |
| שליחת הודעה על קיצור רצועה מהסוף | אורך הודעה | 8 | שם רצועה|זמן להוסיף |
| שליחת הודעה על הוזזת רצועה על ציר הזמן | אורך הודעה | 9 | שם רצועה|זמן התחלה |
| שליחת הודעה על העלאת ווליום לרצועה | אורך הודעה | 10 | שם רצועה |
| שליחת הודעה על הורדת ווליום לרצועה | אורך הודעה | 11 | שם רצועה |
| שליחת הודעה לביצוע undo לרצועה | אורך הודעה | 12 | שם רצועה |
| בקשה להתחיל לערוך רצועה | אורך הודעה | 13 | שם רצועה |
| שליחת הודעה על מחיקת רצועה | אורך הודעה | 14 | שם רצועה |
| הודעה על הפסקת עריכת רצועה | אורך הודעה | 15 | שם רצועה |
| שליחת שם רצועה | אורך הודעה | 16 | שם רצועה |
| שליחת זמן התחלה רצועה | אורך הודעה | 17 | זמן התחלה |
| שליחת ווליום רצועה | אורך הודעה | 18 | ווליום להוסיף\להוריד |
| בקשת שם רצועה | אורך הודעה | 19 | שם רצועה |
| בקשה להשגת רצועות | אורך הודעה | 20 | אין הודעה |

### פרוטוקול שליחת קובץ שרת ולקוח

|  |  |
| --- | --- |
| אורך הודעה 8 ביטים | קובץ הרצועה בבתים |

### צד שרת:

|  |  |  |  |
| --- | --- | --- | --- |
| שם ההודעה | אורך הודעה 5 ביטים | מספר בקשה 2 ביטים | מבנה ההודעה |
| הודעה על שם משתמש/סיסמא לא תקינים | אורך הודעה | 1 | אין הודעה |
| הודעה שהמשתמש מחובר כבר | אורך הודעה | 2 | אין הודעה |
| הודעה על כך ששם המשתמש תפוס | אורך הודעה | 3 | אין הודעה |
| הודעה על כך שההרשמה בוצעה בהצלחה | אורך הודעה | 4 | אין הודעה |
| הודעה על כך שההתחברות בוצעה בהצלחה | אורך הודעה | 5 | אין הודעה |
| שליחת שמות קבצי העריכה של הלקוח | אורך הודעה | 6 | קובץ1; חבר1,חבר2,חבר3|קובץ2;חבר2|קובץ3;חבר2,חבר4,חבר5 |
| הודעה עם כל שמות המשתמשים הקיימים במערכת | אורך הודעה | 7 | משתמש1,משתמש2,משתמש3,משתמש4,משתמש5,משתמש6,משתמש7 |
| הודעה על קובץ עריכה משותף שהתווסף ללקוח | אורך הודעה | 8 | קובץ2; חבר1,חבר2,חבר3 |
| הודעה על משתמש חדש שהצטרף למערכת | אורך הודעה | 9 | משתמש7 |
| שליחת הודעה על קיצור רצועה מההתחלה | אורך הודעה | 10 | שם רצועה|זמן להוריד |
| שליחת הודעה על קיצור רצועה מהסוף | אורך הודעה | 11 | שם רצועה|זמן להוסיף |
| שליחת הודעה על הוזזת רצועה על ציר הזמן | אורך הודעה | 12 | שם רצועה|זמן התחלה |
| שליחת הודעה על העלאת ווליום לרצועה | אורך הודעה | 13 | שם רצועה |
| שליחת הודעה על הורדת ווליום לרצועה | אורך הודעה | 14 | שם רצועה |
| שליחת הודעת נתונים של undo | אורך הודעה | 15 | שם רצועה |
| שליחת הודעה שאין נתונים להחזיר בundo | אורך הודעה | 16 | אין הודעה |
| שליחת הודעה עם שמות החברים שעורכים כעת את הקובץ | אורך הודעה | 17 | משתמש1,משתמש2,משתמש3,משתמש4 |
| שליחת הודעה על רצועה שהתחילו לערוך אותה | אורך הודעה | 18 | שם רצועה|משתמש1 |
| שליחת הודעה על רצועה שהפסיקו לערוך אותה | אורך הודעה | 19 | שם רצועה |
| הודעה על מחיקת רצועה | אורך הודעה | 20 | שם רצועה |
| הודעה ללקוח לפתוח שרת צ'אט | אורך הודעה | 21 | אין הודעה |
| הודעה לחברי הקבוצה מה הip של שרת הצ'אט | אורך הודעה | 22 | כתובת ip |
| שליחת הודעת צ'אט לשאר חברי הקבוצה | אורך הודעה | 23 | הודעת צ'אט |
| שליחת הודעה ללקוח מה הport של שרת הקבצים | אורך הודעה | 24 | מספר port |
| שליחת שם רצועה | אורך הודעה | 25 | שם רצועה |
| שליחת זמן התחלת רצועה | אורך הודעה | 26 | זמן התחלה |
| שליחת ווליום רצועה | אורך הודעה | 27 | ווליום להוסיף |
| שליחת הודעה על מספר רצועות בקובץ | אורך הודעה | 28 | מספר רצועות |
| הודעה שאין קובץ רצועות | אורך הודעה | 29 | אין הודעה |
| שם רצועה תקין | אורך הודעה | 30 | אין הודעות |
| שם רצועה קיים כבר במערכת | אורך הודעה | 31 | אין הודעות |
| שם קובץ קיים לקבוצה | אורך הודעה | 32 | אין הודעה |
| רצועה לא פנויה לעריכה | אורך הודעה | 33 | שם העורך |
| רצועה פנויה לעריכה | אורך הודעה | 34 | אין הודעה |

## תיאור מבני הנתונים

### מסד נתונים:

* מסד נתונים טבלאי שיכיל את שמות המשתמשים ואת הסיסמא המגובבת של כל משתמש
* מסד נתונים טבלאי שיכיל שמות קבצי העריכה המשותפים ונתונים לגבי הרצועות של הקובץ ואת הנתיב למקום בו נשמר קובץ הרצועה

**שם בסיס הנתונים:** MusicEditDB

**שם הטבלה:** users

|  |  |  |
| --- | --- | --- |
| **שם השדה** | username | password |
| **טיפוס** | string | string |
| **דוגמא** | “ofer2405” | “c1f20b2db4dd81221da88074a289bb8e” |

**שם הטבלה:** editFiles

|  |  |  |  |
| --- | --- | --- | --- |
| **שם השדה** | file\_name | group\_members | strips |
| **טיפוס** | string | string | string |
| **דוגמא** | “song2” | “ofer2405,itay700” | “song1;0.1;-10|song2;0.0;30|song4;5.0;-20|song5;0.03;50” |

## סקירת חולשות והאיומים

### שכבת האפליקציה:

* תהליך ה login אימות ווידוא, לכל משתמש יש סיסמא ושם משתמש, לקוח שיכניס נתונים שגויים לא יוכל להיכנס למערכת
* המנעות מ sql INJECTION כתיבת פקודות sql חכמות שלא מאפשרות שתילת קוד לא רצוי כחלק מהפקודות
* ההצפנה היא הצפנה סימטרית והעברת המפתחות מתבצעת על ידי פרוטוקול דיפי הלמן מה שיבטיח העברת מידע בטוחה ברשת וגם מפתח ההצפנה לא יהיה גלוי ברשת
* DOS/DDOS : קשה לבצע התקפת DOS/DDOS במערכת שלי מכיוון שהמערכת שלי היא מערכת יציבה שיכולה להתמודד עם עומס, דאגתי להפריד את התקשורת הכללית ,התקשורת של העברת הקבצים והתקשורת של הצ'אט בפורטים שונים כך שלא יוכל להיווצר עומס במערכת

### שכבת התעבורה:

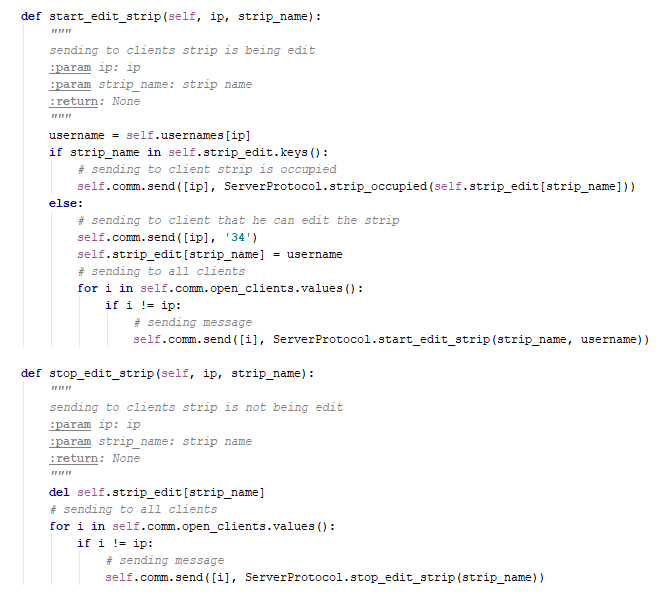
* בחרתי לעבוד בפרוטוקול TCP עם לחיצת יד משולשת כדי להבטיח העברה בטוחה של נתונים ברשת
* כדי להימנע מהתקפות man in the middle דאגתי להצפין את כל המידע שעובר ברשת כך שרק השרת והלקוח בעלי מפתח ההצפנה יוכלו לפענח את המידע

### הפעלת המערכת

* כדי לשמור על עבודה חלקה של המערכת אני אדאג למנוע התקפות dos/ddos בכך שלא אאפשר ללקוח להתחבר לשרת הראשי אם כבר קיים לקוח עם ip זהה לשלו.

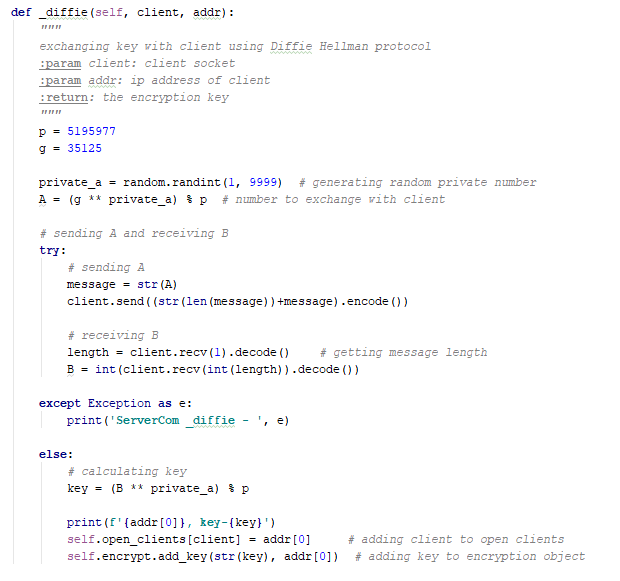
# Umziboom – הקוד

## עבודה במקביל:



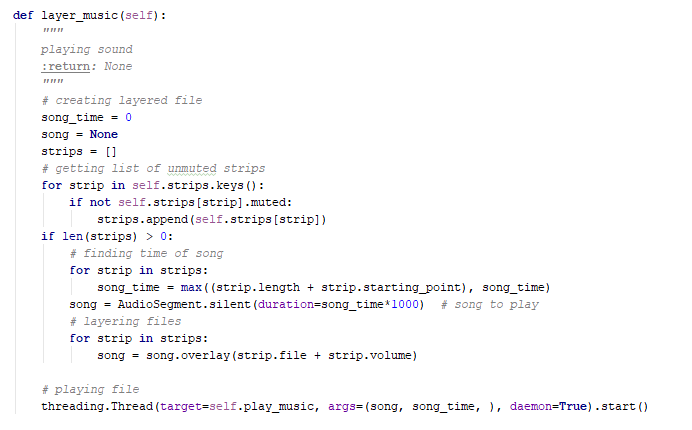
**בקטע קוד ניתן לראות פעולה אחת התחלת עריכת רצועה ובפעולה השניה הפסקת עריכת רצועה. בפרויקט שבחרתי יש עבודה שמתבצעת במקביל. כדי לאפשר את העבודה הזאת מבלי שהמשתמשים ידרסו את העבודות אחד של השני תוך כדי עבודה אני מימשתי את 2 הפעולות האלה בשרת ככה שכל פעם שמשתמש רוצה לערוך רצועה הוא שולח בקשה לשרת והפעולה של השרת בודקת אם המשתמש יכול לערוך את הרצועה ואם מישהו עורך כבר את הרצועה היא שולחת למשתמש הודעה שאין לו אפשרות לערוך. בצורה זו אפשר למנוע לחלוטין התנגשות בין 2 עורכים.**

## החלפת מפתחות דיפי הלמן:



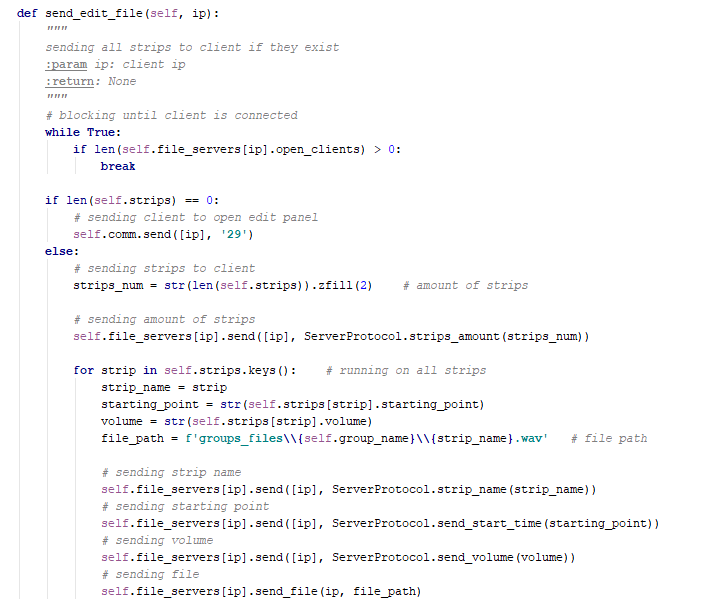
בקטע קוד מעל ניתן לראות פעולה בצד שרת של החלפת מפתחות בשיטת דיפי הלמן כדי לאפשר העברת מפתח להצפנה סימטרית עם הלקוח, הפעולה מאפשרת לבצע החלפת מפתחות ברשת בצורה שהיא לא גלויה ושקשה מאוד לפענח אותה, בצורה זו מתאפשר לנו לבצע תקשורת שהיא מוצפנת ובטוחה ככה שהפרטיות של התקשורת תהיה בטוחה וגם המערכת תהיה בטוחה מהתקפות.

## חיבור שכבות מוזיקה:



פעולת חיבור שכבות המוזיקה היא פעולה\אלגוריתם שבלעדיו הפרויקט שלי לא היה יכול להתקיים. הפעולה מעל היא פעולה שלוקחת את כל הרצועות של הלקוח (כל רצועה זה קובץ מוזיקה שונה) ומחברת אותן לקובץ מוזיקה אחד כדי שבסוף העריכה יהיה אפשר לשמוע שילוב של כל הרצועות הערוכות. בצורה זו ניצן ליצור יצירה מוזיקלית שמורכבת מכמה כלי נגינה וצלילים שונים שמו גיטרה, תופים בס ועוד.

## שליחת קובץ עריכה:



הפעולה מעל היא פעולה שנמצאת בשרת ודואגת לשלוח ללקוח את כל קבצי המוזיקה שלו(הרצועות שלו) עם כל הפרטים שלהם. בפרויקט שלי הקבצים נשמרים לטווח הארוך אצל השרת ולא אצל הלקוח והפעולה דואגת שכל פעם שהלקוח רוצה לערוך קובץ עריכה משותף הוא יקבל את כל קבצי המוזיקה שלו מהשרת ככה שהוא יוכל לעבוד עליהם. הפעולה מתחילה בזה שהיא שולחת ללקוח את כמות הרצועות שהוא צריך לקלוט ורק אז היא רצה על לולאה ששולחת ללקוח את כל הרצועות שלו. בצורת שליחת זו אני יכול להיות בטוח שבלקוח יקבל את כל קבצי המוזיקה שלו בכך שהוא יודע בדיוק כמה רצועות הוא צריך לקלוט.

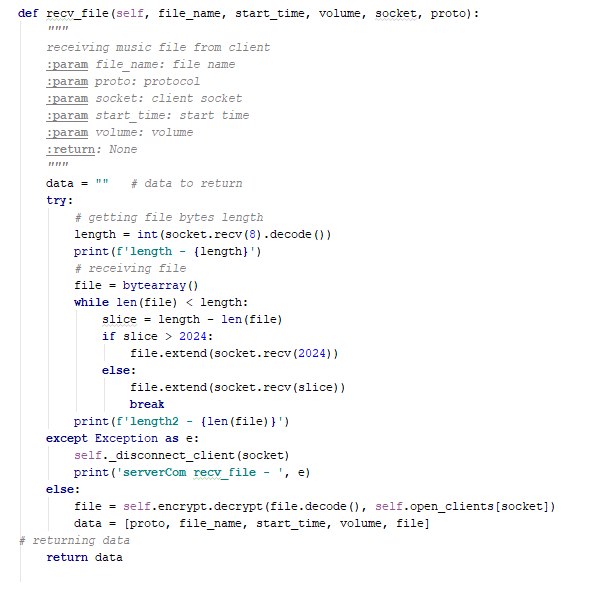
## אתחול רצועות בשרת עריכה:

בפרויקט שלי כל המידע של הרצועות שמור בבסיס הנתונים. כל פעם כשנוצר שרת עריכה הוא צריך לשלוח ללקוחות את כל קבצי הרצועות של הקבוצה, כדי לעשות את זה אצל השרת יש מילון שנקרה strips שהוא מורכב מערך שם רצועה וערך עצם MusicStrip שממנו הוא משיג את המידע על הרצועות. בכל תחילת ריצה של שרת קבוצה המילון strips ריק וכדי לאתחל אותו אני ניגש לבסיס הנתונים בו לכל קובץ עריכה משותף יש עמודה עם מידע על הרצועות שממנה אני משיג את הפרטים על שם הרצועה זמן, ההתחלה שלה והווליום שלה ועם המידע הזה אני מאתחל את הערכים במילון strips.

## התחלת עריכת קובץ עריכה משותף:

פעולות עריכת קובץ משותף היא פעולה שנמצאת בשרת הראשי. בפרויקט שלי יכול להתבצע כמה עריכות של קובץ עריכה משותף במקביל, כדי לפשט את העבודה של השרת אני הפרדתי בין השרת הרגיל לשרת עריכת קובץ ככה שיש לי מחלקת ClientLogic לשרת הראשי ומחלקת EditGroup שאני פותח לכל קבוצה שעורכת קובץ עריכה משותף. כדי להפריד בין בתקשורת של ServerLogic לתקשורת של EditGroup במקום לפתוח חיבור חדש בין השרת ללקוח יצרתי את הפעולה הזאת שמוחקת את הsocket של הלקוח מהopen\_clients של השרת הראשי ומוסיפה אותו לopen\_clients של שרת העריכה ובכך הודעות משרת העריכה לא מגיעות לשרת הראשי ולהפך.

## קליטת קובץ בחלקים:



בפרויקט שלי מתבצעת העברה וקליטה של קבצים בין השרת ללקוח ולהפך. הפעולה למעלה היא פעולת קליטת קובץ בצד השרת. לפני שכתבתי את הפעולה הזאת אני הייתי קולט קבצים בחלקים הרבה יותר גדולים ואז לפעמים כל המידע לא היה מגיע לי לשרת וזה יצר לי באגים. הפעולה דואגת לקלוט את השרת בחלקים קטנים של 2048 ביטים כדי להבטיח שהקובץ שנשלח יגיע בשלמותו מבלי שחלקים יעבדו בדרך ברשת. הפעולה מבטיחה את יציבות המערכת בכך שהיא מנמיכה את הסיכויים שלקוח לא יוכל לערוך את כל הרצועות שלו בגלל שחלק מקבצי הרצועות לא התקבלו בשלמותם.

# Umziboom – בדיקות

## פירוט הבדיקות

| **שם הבדיקה** | **מטרת הבדיקה** | **מה נדרש לבצע** | **מתי** | **מה בוצע בפועל** |
| --- | --- | --- | --- | --- |
| בדיקת יציבות שרת | לבדוק שהשרת יציב והוא לא נופל בשום שלב | מתחברים לשרת עם כמות מרובה של משתמשים ומנסים להתנתק בשלבים שונים בריצה במערכת | 21/5/2023 | * בדיקה ראשונה: התחברתי עם 4 מחשבים למערכת מחשב אחד במסך כניסה מחשב שני במסך בית מחשב שלישי במסך עריכת קובץ ומחשב רביעי אורך רצועה. ניתקתי את 4 המחשבים והשרת לא נפל * בדיקה שניה: ביצעתי את הבדיקה הראשונה והשרת לא נפל |
| בדיקת עריכה של רצועות מוזיקה | לבדוק שאכן יכולת העריכה עובדת | לנסות לערוך רצועת מוזיקה, לשנות את האורך שלה, לשנות ווליום, לשנות את המיקום שלה על ציר הזמן, לקצר רצועה | 21/5/2023 | * בדיקה ראשונה: יצרתי רצועה בקובץ עריכה משותף, שיניתי לה את הווליום, קיצרתי אותה מההתחלה, קיצרתי אותה מהסוף, ואכן השינויים התבצעו על הרצועה(לא יכולתי לבדוק את הזזת מיקום הרצועה כי טרם מימשתי את היכולת) * בדיקה שניה: ביצעתי את הבדיקה הראשונה ואכן השינויים התבצעו על הרצועה. |
| בדיקת התחברות לשרת | לראות שאכן ניתן ליצור התחברות בין לקוח לשרת | לנסות להתחבר לשרת דרך הלקוח ולראות שהוא מקבל את השם משתמש והסיסמה | 21/5/2023 | * בדיקה ראשנה: התחברתי לשרת עם שם משתמש וסיסמא והשרת קיבל את שם המשתמש והסיסמא וחיבר אותי למערכת. * בדיקה שניה: ביצעתי את הבדיקה הראשונה והשרת קיבל את שם המשתמש והסיסמא וחיבר אותי למערכת. |
| הוספת משתמש | לראות שאכן ניתן ליצור משתמש חדש | לנסות ליצור משתמש חדש עם שם משתמש וסיסמה ולראות שהוא אכן מתווסף למערכת | 21/5/2023 | * בדיקה ראשונה: ניסיתי ליצור משתמש חדש עם שם משתמש וסיסמא והוא אכן המשתמש התווסף למערכת * בדיקה שניה: ניסיתי ליצור משתמש חדש עם שם משתמש וסיסמא והוא אכן המשתמש התווסף למערכת |
| יצירת קובץ עריכה משותף | לראות שברגע שיוצרים קובץ עריכה חדש הוא מתווסף גם אצל שאר חברי הקבוצה | לנסות ליצור קובץ עריכה משותף ולבדוק שהוא מתווסף גם אצל שאר חברי הקבוצה | 21/5/2023 | * בדיקה ראשונה: התחברתי עם 2 מחשבים כל מחשב עם משתמש אחר, הוספתי קובץ עריכה משותף למשתמש1 והזמנתי את משתמש2 ומשתמש2 קיבל את קובץ העריכה והוא התווסף אצלו במחשב * בדיקה שניה: ביצעתי את הבדיקה הראשונה ומשתמש2 קיבל את קובץ העריכה והוא התווסף אצלו במחשב |
| סנכרון שינויים | לראות שהשינויים שאתה מבצע בקובץ המוזיקה מתבצעים גם אצל שאר חברי הקבוצה | לנסות לבצע שינויים שונים בקובץ עריכה המשותף ולבדוק שכל השינויים מתבצעים גם אצל כל שאר חברי הקבוצה | 21/5/2023 | * בדיקה ראשונה: התחברתי עם 2 מחשבים לקובץ עריכה משותף, מחשב 1 שינה את האורך ואת הווליום של הרצועה והשינויים התעדכנו במחשב 2 וכשמחשב 2 ביצע את אותם השינויים על הרצועה השינויים התעדכנו במחשב 1 * בדיקה שניה: התחברתי עם 2 מחשבים לקובץ עריכה משותף, מחשב 1 שינה את האורך ואת הווליום של הרצועה והשינויים התעדכנו במחשב 2 וכשמחשב 2 ביצע את אותם השינויים על הרצועה השינויים התעדכנו במחשב 1 |
| בדיקת undo | לבדוק שיכולת undo עובדת | לנסות לבצע undo על רצועה בקובץ עריכה ולראות שאכן הרצועה חוזרת למצב הקודם |  | לא בוצע  ימומש בהמשך |
| הוספה והסרה של רצועות מוזיקה | לבדוק שאכן ניתן להוסיף ולהסיר רצועות מוזיקה | לנסות להוסיף או להסיר רצועות מוזיקה בקובץ העריכה המשותף | 21/5/2023 | * בדיקה ראשונה: התחברתי עם 2 מחשבים לקובץ עריכה משותף, מחשב1 הוסיף רצועה בקובץ והרצועה נוספה אצל מחשב 2(לא מימשתי הסרת רצועה ימומש בהמשך) * בדיקה שניה: התחברתי עם 2 מחשבים לקובץ עריכה משותף, מחשב1 הוסיף רצועה בקובץ והרצועה נוספה אצל מחשב 2(לא מימשתי הסרת רצועה ימומש בהמשך) |
| השמעת קובץ מוזיקה | לבדוק שההשמעה של קובץ המוזיקה עובדת | לנסות להשמיע את קובץ המוזיקה המשותף אצל הלקוח ולנסות להשתיק חלק מהרצועות ואז להשמיע את הקובץ | 21/5/2023 | * בדיקה ראשונה: ניסיתי להשמיע רצועת מוזיקה בקובץ עריכה משותף, השתקתי את חלק מהרצועות ואכן רק הרצועות שלא היו מושתקות נשמעו * בדיקה שניה: ניסיתי להשמיע רצועת מוזיקה בקובץ עריכה משותף, השתקתי את חלק מהרצועות ואכן רק הרצועות שלא היו מושתקות נשמעו |
| שמירת קובץ מוזיקה אצל מחשב לקוח | לבדוק שניתן לשמור את קובץ המוזיקה(שילוב של כל הרצועות) אצל מחשב הלקוח | לנסות לשמור את קובץ העריכה אצל מחשב הלקוח ולראות שהוא נשמר על מחשב הלקוח |  | לא בוצע  ימומש בהמשך |
| זמני תגובה | לבדוק זמני תגובה של המערכת כאשר יש מספר שונה של משתמשים | ליצור קבוצות עם מספר שונה של אנשים או לעבוד כמה קבוצות בו זמנית ולבדוק אם זה משפיע על מהירות זמני התגובה של המערכת ואם נוצר עומס בשרת | 21/5/2023 | * בדיקה ראשונה: התחברתי עם 4 מחשבים, כל 2 מחשבים התחברו לקובץ עריכה משותף, זוג אחד של מחשבים הצליח להתחבר לקובץ העריכה המשותף והזוג השני לא * בדיקה שניה: התחברתי עם 4 מחשבים, כל 2 מחשבים התחברו לקובץ עריכה משותף, זוג אחד של מחשבים הצליח להתחבר לקובץ העריכה המשותף והזוג השני לא |
| אבטחה | לבדוק שרק חברים שהוזמנו יכולים לערוך את הקובץ המשותף | לראות שלאף משתמש אין גישה לקובץ עריכה משותף שהוא לא חלק ממנו | 21/5/2023 | * בדיקה ראשונה: נכנסתי למסך הבית וראיתי שאכן למשתמש שהתחברתי ממנו יש גישה רק לקבצים ששותפו איתו * בדיקה ראשונה: נכנסתי למסך הבית וראיתי שאכן למשתמש שהתחברתי ממנו יש גישה רק לקבצים ששותפו איתו |
| שלמות | לבדוק שהמידע שכל חבר בקבוצת העריכה רואה שלם וזהה לשאר החברים | לנסות לבצע מספר שינויים ועריכות על רצועות המוזיקה בקובץ העריכה המשותף ולבדוק שהשינויים מתבצעים במדויק אצל שאר חברי הקבוצה | 21/5/2023 | * בדיקה ראשונה: התחברתי עם 2 מחשבים לקובץ עריכה משותף, מחשב 1 שינה את האורך ואת הווליום של הרצועה והשינויים התעדכנו במחשב 2 וכשמחשב 2 ביצע את אותם השינויים על הרצועה השינויים התעדכנו במחשב 1   בדיקה שניה: התחברתי עם 2 מחשבים לקובץ עריכה משותף, מחשב 1 שינה את האורך ואת הווליום של הרצועה והשינויים התעדכנו במחשב 2 וכשמחשב 2 ביצע את אותם השינויים על הרצועה השינויים התעדכנו במחשב 1 |

# Umziboom – מדריך למשתמש

## התחברות למערכת:

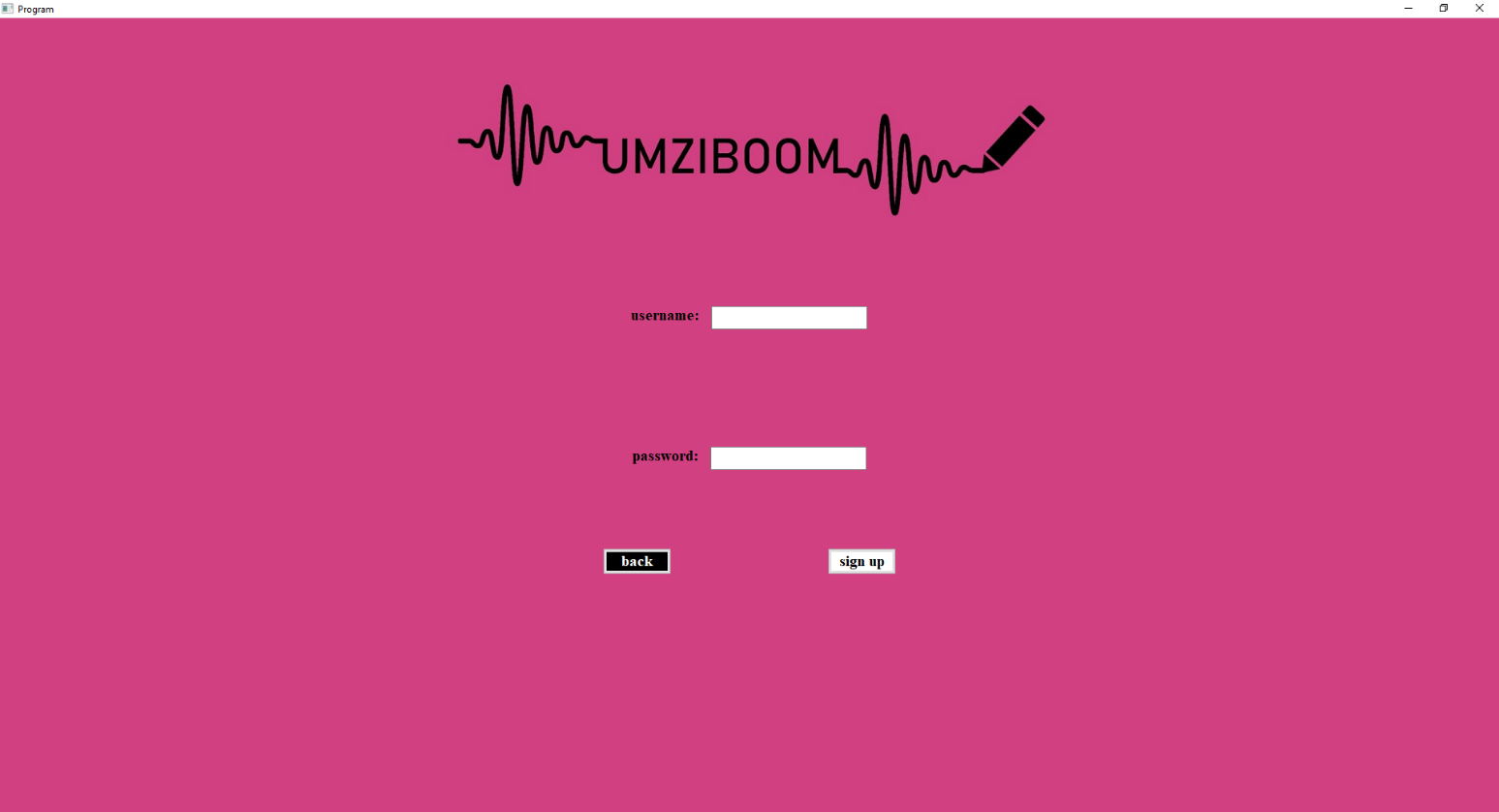
אם רוצים ליצור משתמש חדש לוחצים על הכפתור sign up כדי לעבור למסך ההרשמה

אחרי שמכניסים שם משתמש לוחצים על הכפתור log in כדי להתחבר למערכת ולעבור למסך הבית של המשתמש

מכניסים שם משתמש

מכניסים סיסמא

## הרשמה למערכת:



מכניסים שם משתמש

מכניסים שם משתמש

מכניסים סיסמא

מכניסים סיסמא

לוחצים אם רוצים לחזור למסך התחברות

לוחצים אחרי הכנסת שם משתמש וסיסמא כדי להירשם למערכת

## מסך בית:

תמונה שמכילה טקסט, צילום מסך, גופן, גרפיקה

התיאור נוצר באופן אוטומטי

בחירת קובץ

מכניסים שם משתמש

אחרי שבוחרים קובץ עריכה משותף לוחצים כדי לעבור למסך העריכה

לוחצים אם רוצים לעבור למסך יצירת קובץ עריכה משותף

## מסך יצירת קובץ עריכה משותף:

תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

בוחרים עם איזה משתמשים רוצים לערוך מתוך רשימת המשתמשים הקיימים במערכת

מכניסים את שם קובץ העריכה המשותף

אחרי שבוחרים משתמשים לעריכה ומכניסים שם קובץ עריכה משותף לוחצים כדי ליצור קובץ עריכה משותף חדש ועוברים למסך העריכה

לוחצים אם רוצים לחזור למסך הבית

## מסך עריכת קובץ עריכה משותף:

לוחצים כדי לחזור למסך הבית

לוחצים אם רוצים להוסיף רצועה

לוחצים כדי לראות פרטים של הרצועה

לוחצים כדי לערוך רצועה

לוחצים כדי להשתיק רצועה

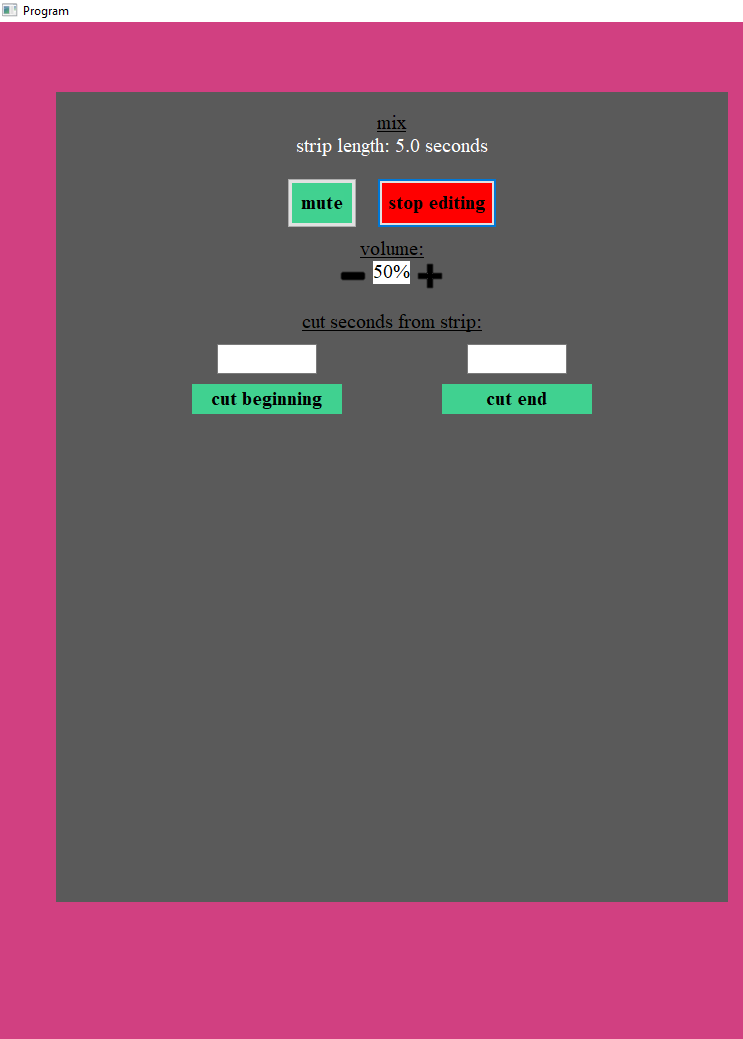
תמונה שמכילה טקסט, צילום מסך, תכונות מולטימדיה, תוכנה גרפית

התיאור נוצר באופן אוטומטי

לוחצים כדי לשמור את קובץ העריכה על המחשב

לוחצים כדי להשמיע מוזיקה

## מסך עריכת קובץ עריכה משותף:



לוחצים כדי לערוך רצועה

לוחצים כדי להנמיך ווליום רצועה

לוחצים כדי להגביר ווליום רצועה

מכניסים שניות לחתוך מתחילת הרצועה

מכניסים שניות לחתוך מסוף הרצועה

אחרי שמכניסים מילישניות לחתוך לוחצים כדי לחתוך את השניות מתחילת הרצועה

אחרי שמכניסים מילישניות לחתוך לוחצים כדי לחתוך את השניות מסוף הרצועה

## העלאת רצועה:

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

מכניסים שם רצועה

אחרי שמכניסים שם רצועה ובוחרים קובץ להעלות לוחצים כדי להוסיף רצועה

לוחצים כדי לחזור למסך עריכה

בוחרים קובץ מוזיקה להעלאה בתור רצועה

לוחצים כדי לפתוח את ה – file explorer

# Umziboom – רפלקציה

במהלך הפרויקט היו לי מספר אתגרים. כבר בתחילת הפרויקט היה לי את האתגר של מציאת דרכים לערוך מוזיקה עם פייטון, חקרתי על הנושא באינטרנט כמה ימים עד שמצאתי את הדרכים שבהם אני יכול לערוך את המוזיקה שבלעדיהם לא הייתי יכול לבצע את הפרויקט. במהלך פיתוח הפרויקט היה בין חודש לחודשיים שאני לא נכחתי בהרבה שיעורי סייבר בגלל הכנות למשלחת בגרמניה והמשלחת עצמה, דבר זה עיקב אותי נורא בתכנון של הקוד ובשלבי העיצוב ובגלל זה אני איחרתי מועדי הגשה של הפרויקט. התמודדתי עם הרבה קשיים, בין הקשיים הכי גדולים שלי היה הלמידה של מודול הגרפיקה wx, לקח לי הרבה מאוד זמן להבין איך משתמשים בו וזה תקע אותי כמה ימים. כדי להתמודד עם הקושי בתכנות הגרפיקה אני נעזרתי בחברים מהמעבדה (עידן גלר ודורון מנשס) והם עזרו לי להבין את הגרפיקה יותר טוב. בתחילת הפרויקט תכננתי לבצע תוכנת עריכה שתכלול גם צ'אט אך בסוף לא נשאר לי זמן כדי לממש את היכולת וגם יכולות עריכה אחרות לא הספקתי לממש אבל עדיין בסוף הצלחתי ליצור פרויקט שעובד ומממש את הרעיון המרכזי שהוא עריכה שיתופית של קבצי מוזיקה. אני רוצה להודות למרי שתמכה בי לאורך כל הפרויקט ושהאמינה בי ואפשרה לי לעשות את הפרויקט הזה אפילו שהוא מאתגר, בנוסף אני רוצה גם להודות על העזרה שהיא נתנה לי לאורך פיתוח הפרויקט ושהיא נתנה לי הזדמנויות לתקן גם כשהגשתי באיחור. אני מודה לאבא שלי שנתן לי את הרעיון לפרויקט והאמין שאני אצליח לבצע אותו. אני מודה לאמא שלי שעזרה לי והייתה שם בשבילי גם כשהיו לי קשיים לאורך הפרויקט. אני מודה לאיתי אוליר, שחר, דין, דניאל, דורון ועידן שעזרו לי בפרויקט בין אם זה היה בהשאלת קטעי קוד או בפתירת באגים בקוד.

# Umziboom – ביבליוגרפיה

עריכת קבצי מוזיקה:

<https://www.geeksforgeeks.org/working-with-wav-files-in-python-using-pydub/>

תיעוד של wxpython:

[https://docs.wxpython.org/wx.lib.intctrl.IntCtrl.html#wx.lib.intctrl.IntCtrl.GetValue](https://docs.wxpython.org/wx.lib.intctrl.IntCtrl.html%23wx.lib.intctrl.IntCtrl.GetValue)

תמונה בתור כפתור עם wxpython:

<https://www.geeksforgeeks.org/wxpython-add-image-in-button/>

מציאת צבעים בהקסדצימלי:  
<https://colorate.azurewebsites.net/Color/8B473D>

שינוי גודל תמונה ב – powerpoint:

[Change the size of a picture, shape, text box, or WordArt - Microsoft Support](https://support.microsoft.com/en-us/office/change-the-size-of-a-picture-shape-text-box-or-wordart-98929cf6-8eab-4d20-87e9-95f2d33c1dde)

איך לשים gif בwxpython:

<https://stackoverflow.com/questions/49389742/animated-gif-with-wxpython-phoenix>

עבירה בין מסכים עם wxpython:

<https://stackoverflow.com/questions/36399627/wxpython-change-panel-by-button>

# נספח א – החקר שבוצע בפרויקט

## client\_comm.py

**import** socket  
**import** threading  
**import** random  
**from** client\_protocol **import** ClientProtocol  
**import** queue  
  
  
**class** ClientCom:  
 *"""  
 client communication  
 """* **def** \_\_init\_\_(self, server\_ip, port, msg\_q, encrypt):  
 *"""  
 initializing parameters and creating communication object* **:param** *server\_ip: server ip* **:param** *port: port of communication* **:param** *msg\_q: queue for messages* **:param** *encrypt: for encrypting and decrypting  
 """* self.running = **True** self.socket = **None** *# server socket* self.server\_ip = server\_ip  
 self.port = port  
 self.msg\_q = msg\_q  
 self.encrypt = encrypt  
  
 *# calling main\_loop thread* threading.Thread(target=self.\_main\_loop, daemon=**True**).start()  
  
 **def** \_main\_loop(self):  
 *"""  
 creating communication with server, exchanging keys with server and starting to get data from server* **:return***: None  
 """* self.socket = socket.socket()  
 **try**:  
 self.socket.connect((self.server\_ip, self.port))  
 **except** Exception **as** e:  
 print(**"client\_comm - \_main\_loop1. "** + str(e))  
 **else**:  
 **if** self.port == 2000:  
 *# calling \_diffie() thread to exchange key with client* key = self.\_diffie()  
 *# putting key in encryption object* self.encrypt.add\_key(key)  
  
 **while** self.running:  
 **try**:  
 *# receiving data from server* len\_data = self.socket.recv(5).decode()  
 data = self.socket.recv(int(len\_data))  
 *# decrypting data* data = self.encrypt.decrypt(data).decode()  
 **if** data[:2] == **'15' or** data[:2] == **'25'**: *# if receive file* proto = int(data[:2]) *# message protocol* file\_name = data[2:] *# file name  
  
 # receiving starting point* data\_len = self.socket.recv(5).decode()  
 start\_time = self.socket.recv(int(data\_len))  
 *# decrypting data* start\_time = self.encrypt.decrypt(start\_time).decode()  
 start\_time = start\_time[2:] *# removing protocol* print(**f'start time - {**start\_time**}'**)  
  
 *# receiving volume* data\_len = self.socket.recv(5).decode()  
 volume = self.socket.recv(int(data\_len))  
 *# decrypting data* volume = self.encrypt.decrypt(volume).decode()  
 volume = volume[2:] *# removing protocol* print(**f'volume - {**volume**}'**)  
  
 *# receiving file* data = self.recv\_file(file\_name, start\_time, volume, proto)  
 **except** Exception **as** e:  
 print(**'ClientComm - main\_loop '**, e)  
 self.msg\_q.put(**"exit"**) *# to end main client code* **break  
 else**:  
 **if** type(data) == str: *# if not receive file* data = ClientProtocol.unpack(data)  
 self.msg\_q.put(data)  
  
 **def** \_diffie(self):  
 *"""  
 exchanging key with client using Diffie Hellman protocol* **:param** *client: client socket* **:param** *addr: ip address of client* **:return***: the encryption key  
 """* key = -1  
 p = 5195977  
 g = 35125  
  
 private\_b = random.randint(1, 9999) *# generating random private number* B = (g \*\* private\_b) % p *# number to exchange with client  
  
 # sending B and receiving A* **try**:  
 *# sending B* message = str(B)  
 self.socket.send((str(len(message)) + message).encode())  
  
 *# receiving B* length = self.socket.recv(1).decode() *# getting message length* A = int(self.socket.recv(int(length)).decode())  
  
 **except** Exception **as** e:  
 print(**'ClientComm - \_diffie '**, e)  
  
 **else**:  
 *# calculating key* key = (A \*\* private\_b) % p  
 print(**f'key-{**key**}'**)  
 **return** str(key)  
  
 **def** send(self, msg):  
 *"""  
 sending messages to server* **:param** *msg: message to send* **:return***: None  
 """* **try**:  
 enc = self.encrypt.encrypt(msg)  
 *# sending length* self.socket.send(str(len(enc)).zfill(5).encode())  
 *# sending encrypted message* self.socket.send(enc)  
 **except** Exception **as** e:  
 print(**'ClientCom send - '**, e)  
  
 **def** send\_file(self, file\_path):  
 *"""  
 sending file to server* **:param** *file\_path: file path* **:return***: None  
 """  
 # getting file* **with** open(file\_path, **'rb'**) **as** strip:  
 file = strip.read()  
 enc = self.encrypt.encrypt(file)  
  
 length = str(len(enc)).zfill(8)  
 print(**f'length1 - {**length**}'**)  
 **try**:  
 self.socket.send(str(len(enc)).zfill(8).encode()) *# sending file length* **except** Exception **as** e:  
 print(**'ClientCom send\_file - '**, e)  
 **else**:  
 self.socket.sendall(enc)  
  
 **def** recv\_file(self, file\_name, start\_time, volume, proto):  
 *"""  
 receiving file from server* **:param** *file\_name: file name* **:param** *start\_time: start time* **:param** *volume: volume* **:param** *proto: protocol number* **:return***: None  
 """* data = **""** *# data to return* **try**:  
 *# getting file bytes length* length = int(self.socket.recv(8).decode())  
 print(**f'length - {**length**}'**)  
 *# receiving file* file = bytearray()  
 **while** len(file) < length:  
 slice = length - len(file)  
 **if** slice > 2048:  
 file.extend(self.socket.recv(2048))  
 **else**:  
 file.extend(self.socket.recv(slice))  
 **break** print(**f'length2 - {**len(file)**}'**)  
 **except** Exception **as** e:  
 print(**'clientCom recv\_file - '**, e)  
 **else**:  
 file = self.encrypt.decrypt(file.decode())  
 data = [proto, file\_name, start\_time, volume, file]  
 *# returning data* **return** data

## client\_Logic.py

**from** client\_com **import** ClientCom  
**from** encrypt **import** Encrypt  
**import** queue  
**import** threading  
**from** gui **import**\*  
**import** wx  
**from** pubsub **import** pub  
**from** music\_strip **import** MusicStrip  
**import** os  
**from** pydub **import** AudioSegment  
**from** pydub.playback **import** \_play\_with\_simpleaudio  
**import** time  
  
  
**class** ClientLogic:  
 *"""client logic"""* **def** \_\_init\_\_(self):  
 *# queue for messages* self.msg\_q = queue.Queue()  
 *# queue for file messages* self.file\_q = queue.Queue()  
 *# encrypt object* self.cp = Encrypt()  
 *# music strips {strip name: MusicStrip}* self.strips = {}  
 *# client comm* self.comm = ClientCom(**'192.168.1.21'**, 2000, self.msg\_q, self.cp)  
 *# file comm* self.file\_comm = **None** *# strips counter to receive strips* self.strips\_count = 0  
 *# simple audio play* self.play = **None** *# calling handle msgs* threading.Thread(target=self.handle\_msgs, daemon=**True**).start()  
  
 **def** handle\_msgs(self):  
 *"""  
 handling msgs from server* **:return***: None  
 """* **while True**:  
 msg = self.msg\_q.get()  
 **if** msg[0] == 1: *# username or password incorrect  
 # telling pasword or username are incorrect* wx.CallAfter(pub.sendMessage, **"fail\_login"**, msg=**'password or username are incorrect'**)  
 **elif** msg[0] == 2: *# user already logged in* wx.CallAfter(pub.sendMessage, **"user\_already\_connected"**, msg=**'user is already connected'**)  
 **elif** msg[0] == 3: *# sign in fail* wx.CallAfter(pub.sendMessage, **"fail\_sign\_in"**, msg=**'user already exists'**)  
 **elif** msg[0] == 4: *# sign in successful* wx.CallAfter(pub.sendMessage, **"success\_sign\_in"**, event=**None**)  
 **elif** msg[0] == 5: *# login successful* wx.CallAfter(pub.sendMessage, **"success\_login"**, event=**None**)  
 **elif** msg[0] == 6: *# receive edit file names* wx.CallAfter(pub.sendMessage, **"set\_files"**, file\_names=msg[1:])  
 **elif** msg[0] == 7:  
 wx.CallAfter(pub.sendMessage, **"set\_users"**, users=msg[1].split(**','**))  
 **elif** msg[0] == 8: *# add file* wx.CallAfter(pub.sendMessage, **"add\_file"**, file\_name=msg[1])  
 **elif** msg[0] == 9: *# added user* wx.CallAfter(pub.sendMessage, **"add\_user"**, username=msg[1])  
 **elif** msg[0] == 10: *# cut\_beginning* self.cut\_beginning(msg[1], msg[2])  
 wx.CallAfter(pub.sendMessage, **"cut\_beginning"**, strip\_name=msg[1], cut\_time=float(msg[2]), flag=**False**)  
 **elif** msg[0] == 11: *# cut end* self.cut\_end(msg[1], msg[2])  
 wx.CallAfter(pub.sendMessage, **"cut\_end"**, strip\_name=msg[1], cut\_time=float(msg[2]), flag=**False**)  
 **elif** msg[0] == 12:  
 **pass  
 elif** msg[0] == 13: *# inc volume* self.inc(msg[1])  
 wx.CallAfter(pub.sendMessage, **"inc\_volume"**, event=msg[1])  
 **elif** msg[0] == 14: *# dec volume* self.dec(msg[1])  
 wx.CallAfter(pub.sendMessage, **"dec\_volume"**, event=msg[1])  
 **elif** msg[0] == 15:  
 **pass  
 elif** msg[0] == 16:  
 **pass  
 elif** msg[0] == 17:  
 **pass  
 elif** msg[0] == 18: *# editor started editing* wx.CallAfter(pub.sendMessage, **"start\_stop\_edit"**, strip\_name=msg[1], username=msg[2], edit=**True**)  
 **elif** msg[0] == 19: *# editor stopped editing* wx.CallAfter(pub.sendMessage, **"start\_stop\_edit"**, strip\_name=msg[1], username=**''**, edit=**False**)  
 **elif** msg[0] == 20:  
 **pass  
 elif** msg[0] == 21:  
 **pass  
 elif** msg[0] == 22:  
 **pass  
 elif** msg[0] == 23:  
 **pass  
 elif** msg[0] == 24: *# open file client* port = int(msg[1])  
 *# opening a file client* self.file\_comm = ClientCom(**'192.168.1.21'**, port, self.file\_q, self.cp)  
 *# sending file\_comm to gui* wx.CallAfter(pub.sendMessage, **"file\_comm"**, file\_comm=self.file\_comm)  
 *# starting handle files thread* threading.Thread(target=self.handle\_files, daemon=**True**).start()  
  
 **elif** msg[0] == 26:  
 **pass  
 elif** msg[0] == 27:  
 **pass  
 elif** msg[0] == 29: *# you have no strips* wx.CallAfter(pub.sendMessage, **"open\_edit"**, event=**None**)  
 **elif** msg[0] == 30: *# strip name is ok* wx.CallAfter(pub.sendMessage, **"send\_strip"**)  
 **elif** msg[0] == 31: *# strip already exists* wx.CallAfter(pub.sendMessage, **"strip\_exists"**, msg=**'strip name already exists in your edit file'**)  
 **elif** msg[0] == 32: *# file already exists* wx.CallAfter(pub.sendMessage, **"file\_exists"**, msg=**'file already exists'**)  
 **elif** msg[0] == 33: *# strip occupied* wx.CallAfter(pub.sendMessage, **"strip\_occupied"**, msg=**f'{**msg[1]**} is already editing this strip'**)  
 **elif** msg[0] == 34: *# edit strip* wx.CallAfter(pub.sendMessage, **"edit\_strip"**)  
  
 **def** handle\_files(self):  
 *"""  
 handling msgs from file server* **:return***: None  
 """* **while True**:  
 msg = self.file\_q.get()  
  
 **if** msg[0] == 25: *# add strip* self.add\_strip(msg[1], float(msg[2]), int(msg[3]), msg[4])  
 **if** self.strips\_count > 1:  
 self.strips\_count -= 1  
 **elif** self.strips\_count != 0:  
 *# showing edit panel* wx.CallAfter(pub.sendMessage, **"open\_edit"**, event=**None**)  
 self.strips\_count = 0  
 **elif** msg[0] == 28: *# amount of strips to receive  
 # show loading panel* self.strips\_count = int(msg[1])  
 wx.CallAfter(pub.sendMessage, **"loading"**, event=**None**)  
  
 **def** add\_strip(self, strip\_name, starting\_point, volume, file\_path):  
 *"""  
 creates MusicStrip object* **:param** *file\_path: file path* **:param** *strip\_name: strip name* **:param** *starting\_point: starting point* **:param** *volume: volume* **:return***: None  
 """* **if** type(file\_path) == str:  
 *# opening file* **with** open(file\_path, **'rb'**) **as** strip:  
 file = strip.read()  
 **else**:  
 file = file\_path  
 self.strips[strip\_name] = MusicStrip(file, strip\_name, starting\_point, volume, **'client\_music'**)  
 length = self.strips[strip\_name].length  
  
 *# setting strip in gui* wx.CallAfter(pub.sendMessage, **"add\_strip"**, strip\_name=strip\_name, starting\_point=starting\_point, volume=volume  
 , length=length)  
  
 **def** layer\_music(self):  
 *"""  
 playing sound* **:return***: None  
 """  
 # creating layered file* song\_time = 0  
 song = **None** strips = []  
 *# getting list of unmuted strips* **for** strip **in** self.strips.keys():  
 **if not** self.strips[strip].muted:  
 strips.append(self.strips[strip])  
 **if** len(strips) > 0:  
 *# finding time of song* **for** strip **in** strips:  
 song\_time = max((strip.length + strip.starting\_point), song\_time)  
 song = AudioSegment.silent(duration=song\_time\*1000) *# song to play  
 # layering files* **for** strip **in** strips:  
 song = song.overlay(strip.file + strip.volume)  
  
 *# playing file* threading.Thread(target=self.play\_music, args=(song, song\_time, ), daemon=**True**).start()  
  
 **def** play\_music(self, song, song\_time):  
 *"""  
 playing music* **:param** *song: song to play* **:param** *song\_time: song time* **:return***: None  
 """* self.play = **None  
 if** song:  
 self.play = \_play\_with\_simpleaudio(song)  
 time.sleep(song\_time) *# do some stuff inbetween* self.stop\_play\_music()  
  
 **def** stop\_play\_music(self):  
 *"""  
 stop playing music* **:return***: None  
 """* **if** self.play:  
 self.play.stop()  
 wx.CallAfter(pub.sendMessage, **"music\_stopped"**, event=**None**)  
  
 **def** mute\_unmute(self, strip\_name):  
 *"""  
 mutting or un mutting strip* **:param** *strip\_name: strip name* **:return***: None  
 """* **if** self.strips[strip\_name].muted:  
 self.strips[strip\_name].muted = **False  
 else**:  
 self.strips[strip\_name].muted = **True  
  
 def** inc(self, strip\_name):  
 *"""  
 increasing volume on strips* **:param** *strip\_name: strip name* **:return***: None  
 """* self.strips[strip\_name].volume += 10  
  
 **def** dec(self, strip\_name):  
 *"""  
 decreasing volume on strips* **:param** *strip\_name: strip name* **:return***: None  
 """* self.strips[strip\_name].volume -= 10  
  
 **def** cut\_beginning(self, strip\_name, time):  
 *"""  
 cutting from beginning* **:param** *strip\_name: strip\_name* **:param** *time: time to cut* **:return***: None  
 """* self.strips[strip\_name].trim\_start(float(time))  
  
 **def** cut\_end(self, strip\_name, time):  
 *"""  
 cutting from end* **:param** *strip\_name: strip\_name* **:param** *time: time to cut* **:return***: None  
 """* self.strips[strip\_name].trim\_end(float(time))  
  
 **def** graphic\_loop(self):  
 *"""  
 starting graphics* **:return***: None  
 """* app = wx.App(**False**)  
 frame = Program(self.comm)  
 frame.Show()  
 app.MainLoop()  
  
  
**def** main():  
 logic = ClientLogic() *# logic object  
 # pubsubs* pub.subscribe(logic.add\_strip, **"add\_strip\_logic"**)  
 pub.subscribe(logic.layer\_music, **"play\_music"**)  
 pub.subscribe(logic.stop\_play\_music, **"stop\_play"**)  
 pub.subscribe(logic.mute\_unmute, **"mute"**)  
 pub.subscribe(logic.inc, **"inc"**)  
 pub.subscribe(logic.dec, **"dec"**)  
 pub.subscribe(logic.cut\_beginning, **"beginning"**)  
 pub.subscribe(logic.cut\_end, **"end"**)  
  
 *# graphic loop* logic.graphic\_loop()  
  
 *# deleting all files from client\_music directory* path = **'client\_music'  
  
 for** f **in** os.listdir(path):  
 os.remove(os.path.join(path, f))  
  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

## client\_protocol.py

**class** ClientProtocol:  
 *"""  
 class to build messages based on client protocol  
 """* @staticmethod  
 **def** log\_in(username, password):  
 *"""  
 creates log in message* **:param** *username: username* **:param** *password: password* **:return***: message based on protocol  
 """* message = **'01'**+password+**'|'**+username  
 **return** message  
  
 @staticmethod  
 **def** sign\_in(username, password):  
 *"""  
 ign in message* **:param** *username: username* **:param** *password: password* **:return***: message based on protocol  
 """* message = **'02'** + password + **'|'** + username  
 **return** message  
  
 @staticmethod  
 **def** create\_file(file\_name):  
 *"""  
 create file message* **:param** *file\_name: file name* **:return***: message based on protocol  
 """* message = **'03'** + file\_name  
 **return** message  
  
 @staticmethod  
 **def** start\_edit\_file(file\_name):  
 *"""  
 start edit file message* **:param** *file\_name: file name* **:return***: message based on protocol  
 """* message = **'04'** + file\_name  
 **return** message  
  
 @staticmethod  
 **def** shorten\_start(strip\_name, time\_to\_take):  
 *"""  
 shorten start messsage* **:param** *strip\_name: strip name* **:param** *time\_to\_take: time to take* **:return***: message based on protocol  
 """* message = **'07'** + strip\_name + **'|'** + str(time\_to\_take)  
 **return** message  
  
 @staticmethod  
 **def** shorten\_end(strip\_name, time\_to\_take):  
 *"""  
 shorten end messsage* **:param** *strip\_name: strip name* **:param** *time\_to\_take: time to take* **:return***: message based on protocol  
 """* message = **'08'** + strip\_name + **'|'** + str(time\_to\_take)  
 **return** message  
  
 @staticmethod  
 **def** move\_strip(strip\_name, start\_time):  
 *"""  
 move strip message* **:param** *strip\_name: strip name* **:param** *start\_time: start time* **:return***: message based on protocol  
 """* message = **'09'** + strip\_name + **'|'** + str(start\_time)  
 **return** message  
  
 @staticmethod  
 **def** inc\_volume(strip\_name):  
 *"""  
 increase volume message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '10'** + strip\_name  
  
 @staticmethod  
 **def** dec\_volume(strip\_name):  
 *"""  
 decrease volume message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '11'** + strip\_name  
  
 @staticmethod  
 **def** undo(strip\_name):  
 *"""  
 undo message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '12'** + strip\_name  
  
 @staticmethod  
 **def** start\_edit\_strip(strip\_name):  
 *"""  
 start edit strip message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '13'** + strip\_name  
  
 @staticmethod  
 **def** delete\_strip(strip\_name):  
 *"""  
 delete strip message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '14'** + strip\_name  
  
 @staticmethod  
 **def** stop\_editing\_strip(strip\_name):  
 *"""  
 send stop editing strip message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '15'** + strip\_name  
  
 @staticmethod  
 **def** add\_strip(strip\_name):  
 *"""  
 send strip name message* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '16'** + strip\_name  
  
 @staticmethod  
 **def** send\_start\_time(start\_time):  
 *"""  
 send start time* **:param** *start\_time: start time* **:return***: message based on protocol  
 """* **return '17'** + start\_time  
  
 @staticmethod  
 **def** send\_volume(volume):  
 *"""  
 send start time* **:param** *volume: volume* **:return***: message based on protocol  
 """* **return '18'** + volume  
  
 @staticmethod  
 **def** strip\_name\_request(strip\_name):  
 *"""  
 send strip name* **:param** *strip\_name: strip name* **:return***: message based on protocol  
 """* **return '19'** + strip\_name  
  
 @staticmethod  
 **def** unpack(msg):  
 *"""  
 unpacks messages from server* **:param** *msg: msg in protocol* **:return***: list of opcode params  
 """* return\_list = [] *# list to return  
  
 # number of protocol* proto\_num = int(msg[:2])  
  
 *# unpack* **if '|' not in** msg:  
 return\_list = [proto\_num, msg[2:]]  
  
 **else**:  
 return\_list = [proto\_num] + msg[2:].split(**'|'**)  
  
 **return** return\_list  
  
  
  
**def** main():  
 **pass  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

## data\_base.py

**import** sqlite3  
  
  
**class** DB:  
 *"""  
 class to manage data base  
 """* **def** \_\_init\_\_(self):  
 *"""  
 creating DB object  
 """* self.db\_name = **'MusicEditDB.db'** *# data base name* self.users\_table = **'users'** *# users table name* self.editors\_table = **'editFiles'** *# editors table name* self.conn = **None** self.cursor = **None** *# creating data bass and tables* self.\_create\_db()  
  
 **def** \_create\_db(self):  
 *"""  
 creates data bass and tables if they don't exist* **:return***: None  
 """  
  
 # connecting to data base* self.conn = sqlite3.connect(self.db\_name)  
   
 *# creating cursor* self.cursor = self.conn.cursor()  
  
 *# creating users table if not exists* self.cursor.execute(**f'''CREATE TABLE IF NOT EXISTS {**self.users\_table**}   
 (username text, password text)'''**)  
  
 *# creating editFiles table if not exists* self.cursor.execute(**f'''CREATE TABLE IF NOT EXISTS {**self.editors\_table**}   
 (file\_name text, group\_members text, strips text)'''**)  
  
 self.conn.commit()  
  
 **def** \_username\_exists(self, username):  
 *"""  
 checks if username exists* **:param** *username: username* **:return***: if username exists  
 """* sql = **f"SELECT \* FROM {**self.users\_table**} WHERE username = ?"** self.cursor.execute(sql, (username,))  
  
 **return** self.cursor.fetchone()  
  
 **def** add\_user(self, username, password):  
 *"""  
 adds user if he does'nt already exists* **:param** *username: username* **:param** *password: password* **:return***: if username was added or not  
 """* status = **False** *# if username doesn't exist* **if not** self.\_username\_exists(username):  
 sql = **f"INSERT INTO {**self.users\_table**} VALUES (?,?)"** self.cursor.execute(sql, (username, password,)) *# adding username and password* self.conn.commit()  
 status = **True  
  
 return** status  
  
 **def** check\_password(self, username, password):  
 *"""  
 checks if password is correct* **:param** *username: username* **:param** *password: password* **:return***: if password is correct  
 """* sql = **f"SELECT password FROM {**self.users\_table**} WHERE username = ?"** self.cursor.execute(sql, (username,))  
 password\_got = self.cursor.fetchone()  
 **return** password\_got **and** password\_got[0] == password  
  
 **def** \_file\_exists(self, file\_name, group\_members):  
 *"""  
 checks if file name already exists for the group* **:param** *file\_name: file name* **:param** *group\_members: group editors* **:return***: if file already exists  
 """* group\_members = group\_members.split(**','**)  
 ret = **False** sql = **f"SELECT group\_members FROM {**self.editors\_table**} WHERE file\_name = ?"** self.cursor.execute(sql, (file\_name,))  
  
 *# getting files of group* editor\_names = self.cursor.fetchall()  
 print(editor\_names)  
  
 *# running on file names* **for** i **in** editor\_names:  
 **if** ret:  
 **break  
 for** name **in** group\_members:  
 **if** name **not in** i[0].split(**','**):  
 ret = **False  
 break** ret = **True  
  
 return** ret  
  
 **def** add\_file(self, file\_name, group\_members):  
 *"""  
 adds file if it doesn't already exist* **:param** *file\_name: file name* **:param** *group\_members: group members* **:return***: if file was added  
 """* status = **False** *# checking if file already exists* **if not** self.\_file\_exists(file\_name, group\_members):  
 sql = **f"INSERT INTO {**self.editors\_table**} VALUES (?,?,?)"** self.cursor.execute(sql, (file\_name, group\_members, **None**,)) *# adding file name and group members* self.conn.commit()  
 status = **True  
  
 return** status  
  
 **def** update\_file(self, file\_name, group\_members, details):  
 *"""  
 updates strips content* **:param** *file\_name: file name* **:param** *group\_members: group members* **:param** *details: details to update* **:return***: None  
 """* sql = **f"UPDATE {**self.editors\_table**} SET strips = ? WHERE file\_name = ? AND group\_members = ?"** self.cursor.execute(sql, (details, file\_name, group\_members,)) *# adding file name and group members* self.conn.commit()  
  
 **def** get\_details(self, file\_name, group\_members):  
 *"""  
 gets details of file strips* **:param** *file\_name: file name* **:param** *group\_members: group members* **:return***: details of file strips  
 """* sql = **f"SELECT strips FROM {**self.editors\_table**} WHERE file\_name = ? AND group\_members = ?"** self.cursor.execute(sql, (file\_name, group\_members,))  
 **return** self.cursor.fetchone()[0]  
  
 **def** get\_users(self, username):  
 *"""  
 gets all names of users* **:param** *username: username to exclude* **:return***: names of all users  
 """* sql = **f"SELECT username FROM {**self.users\_table**} WHERE username != ?"** self.cursor.execute(sql, (username, ))  
 **return** [x[0] **for** x **in** self.cursor.fetchall()]  
  
 **def** get\_user\_files(self, username):  
 *"""  
 gets all the files of user* **:param** *username: username* **:return***: None  
 """* edit\_files = []  
 sql = **f"SELECT file\_name, group\_members FROM {**self.editors\_table**}"** self.cursor.execute(sql)  
 ans = self.cursor.fetchall()  
 *# finding edit files of username* **for** i **in** ans:  
 **if** username **in** i[1].split(**','**):  
 edit\_files.append(**';'**.join(i))  
 **return** edit\_files  
  
 **def** close\_db(self):  
 *"""  
 closes data base connection* **:return***: None  
 """* self.conn.close()  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 **pass**

## edit\_group.py

**from** server\_com **import** ServerCom  
**from** edit\_group\_com **import** EditGroupCom  
**import** queue  
**from** encrypt **import** Encrypt  
**from** data\_base **import** DB  
**import** threading  
**from** server\_protocol **import** ServerProtocol  
**from** music\_strip **import** MusicStrip  
**import** time  
**import** os  
  
  
**class** EditGroup:  
 *"""class server for file editing with group"""* **def** \_\_init\_\_(self, socket, username, group\_name, server\_q, encrypt, ip):  
 *"""  
 creating edit group object* **:param** *socket: first client socket* **:param** *ip: first client ip* **:param** *username: first client username* **:param** *group\_name: group name  
 """  
 # running flag* self.running = **True** *# port for file servers* self.file\_port = 2001  
 *# encrypt object* self.encrypt = encrypt  
 *# group name* self.group\_name = group\_name  
 *# queue for messages from clients* self.msg\_q = queue.Queue()  
 *# q to send messages to main server* self.server\_q = server\_q  
 *# dictionary of msg queues for file servers {ip, queue}* self.file\_queues = {ip: queue.Queue()}  
 *# dictionary of strips {strip name : MusicStrip object}* self.strips = {}  
 *# {strip\_name:[list of changes]}* self.changes = {}  
 *# dictionary of file servers {ip : file\_server}* self.file\_servers = {}  
 *# creating edit group communication* self.comm = EditGroupCom(self.msg\_q, self.encrypt, socket, ip)  
 *# usernames = {ip, username}* self.usernames = {ip: username}  
 *# dict {strip name : username}* self.strip\_edit = {}  
  
 *# setting strips* self.set\_strips()  
 *# creating file server for client* self.file\_servers[ip] = ServerCom(self.file\_port, self.file\_queues[ip], self.encrypt)  
 *# sending client port to open a file client* self.comm.send([ip], ServerProtocol.file\_server\_port(str(self.file\_port)))  
 *# sending to client details about file* self.send\_edit\_file(ip)  
 *# handle msgs* threading.Thread(target=self.handle\_msgs).start()  
 *# handle files for first client* threading.Thread(target=self.handle\_files, args=(self.file\_queues[ip],)).start()  
  
 **def** handle\_msgs(self):  
 *"""  
 handling msgs from server* **:return***: None  
 """* **while** self.running:  
 ip, msg = self.msg\_q.get()  
  
 **if** msg[0] == -1: *# close edit group  
 # saving file* self.save\_file()  
 self.server\_q.put((ip, [-1, self.group\_name]))  
 self.running = **False  
 elif** msg[0] == 0:  
 *# removing client* self.remove\_client\_editing(ip)  
 **del** self.usernames[ip]  
 *# closing client file server* self.file\_servers[ip].running = **False  
 del** self.file\_servers[ip]  
 **del** self.file\_queues[ip]  
 *# telling main server to remove client* self.server\_q.put((ip, [0]))  
  
 **elif** msg[0] == 5:  
 **pass  
 elif** msg[0] == 6:  
 **pass  
 elif** msg[0] == 7: *# cut from beginning* self.cut\_beginning(ip, msg[1], msg[2])  
 **elif** msg[0] == 8: *# cut from end* self.cut\_end(ip, msg[1], msg[2])  
 **elif** msg[0] == 9:  
 **pass  
 elif** msg[0] == 10: *# inc volume* self.inc(ip, msg[1])  
 **elif** msg[0] == 11: *# dec volume* self.dec(ip, msg[1])  
 **elif** msg[0] == 12:  
 **pass  
 elif** msg[0] == 13: *# start editing strip* self.start\_edit\_strip(ip, msg[1])  
 **elif** msg[0] == 14:  
 **pass  
 elif** msg[0] == 15: *# stop editing strip* self.stop\_edit\_strip(ip, msg[1])  
 **elif** msg[0] == 16:  
 **pass  
 elif** msg[0] == 17:  
 **pass  
 elif** msg[0] == 18:  
 **pass  
 elif** msg[0] == 19:  
 self.check\_strip\_exists(ip, msg[1])  
  
 **def** handle\_files(self, file\_q):  
 *"""  
 handling file msgs from client* **:return***: None  
 """* **while** self.running:  
 ip, msg = file\_q.get()  
  
 **if** msg[0] == 16: *# add strip* self.add\_strip(msg[1], msg[2], msg[3], msg[4], ip)  
  
 **def** add\_editor(self, socket, username, ip):  
 *"""  
 adding socket to open clients* **:param** *socket: client socket* **:param** *ip: client ip* **:param** *username: username* **:return***: None  
 """* self.file\_port += 1  
 *# adding client* self.comm.open\_clients[socket] = ip  
 self.usernames[ip] = username  
 self.file\_queues[ip] = queue.Queue()  
 *# creating file server for client* self.file\_servers[ip] = ServerCom(self.file\_port, self.file\_queues[ip], self.encrypt)  
 *# sending client port to open a file client* self.comm.send([ip], ServerProtocol.file\_server\_port(str(self.file\_port)))  
 *# sending to client details about file* self.send\_edit\_file(ip)  
  
 *# sending all strip editors* **for** strip **in** self.strip\_edit.keys():  
 self.comm.send([ip], ServerProtocol.start\_edit\_strip(strip, self.strip\_edit[strip]))  
  
 *# handle files for first client* threading.Thread(target=self.handle\_files, args=(self.file\_queues[ip],)).start()  
  
 **def** inc(self, ip, strip\_name):  
 *"""  
 increasing volume and sending to client* **:param** *strip\_name: strip name* **:param** *ip: client ip* **:return***: None  
 """  
 # increasing volume* self.strips[strip\_name].volume += 10  
 *# sending editors to inc volume* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 self.comm.send([i], ServerProtocol.inc\_volume(strip\_name))  
  
 **def** dec(self, ip, strip\_name):  
 *"""  
 decreasing volume and sending to client* **:param** *strip\_name: strip name* **:param** *ip: client ip* **:return***: None  
 """  
 # decreasing volume* self.strips[strip\_name].volume -= 10  
 *# sending editors to dec volume* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 self.comm.send([i], ServerProtocol.dec\_volume(strip\_name))  
  
 **def** add\_strip(self, strip\_name, start\_time, volume, file, ip):  
 *"""  
 adds sends to users to add strips* **:param** *strip\_name: strip name* **:param** *start\_time: start time* **:param** *volume: volume* **:param** *file: file in bytes* **:param** *ip: ip to exclude* **:return***: None  
 """* file\_path = **f'groups\_files\\{**self.group\_name**}\\{**strip\_name**}.wav'** *# file path  
 # saving strip file* **with** open(file\_path, **'wb'**) **as** strip:  
 strip.write(file)  
  
 *# adding strip to strips list* self.strips[strip\_name] = MusicStrip(**None**, strip\_name, float(start\_time)  
 , int(volume), **f'groups\_files\\{**self.group\_name**}'**)  
  
 *# sending added strip to connected clients* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 *# sending strip name* self.file\_servers[i].send([i], ServerProtocol.strip\_name(strip\_name))  
 *# sending starting point* self.file\_servers[i].send([i], ServerProtocol.send\_start\_time(start\_time))  
 *# sending volume* self.file\_servers[i].send([i], ServerProtocol.send\_volume(volume))  
 *# sending file* self.file\_servers[i].send\_file(i, file\_path)  
  
 **def** send\_edit\_file(self, ip):  
 *"""  
 sending all strips to client if they exist* **:param** *ip: client ip* **:return***: None  
 """  
 # blocking until client is connected* **while True**:  
 **if** len(self.file\_servers[ip].open\_clients) > 0:  
 **break  
  
 if** len(self.strips) == 0:  
 *# sending client to open edit panel* self.comm.send([ip], **'29'**)  
 **else**:  
 *# sending strips to client* strips\_num = str(len(self.strips)).zfill(2) *# amount of strips  
  
 # sending amount of strips* self.file\_servers[ip].send([ip], ServerProtocol.strips\_amount(strips\_num))  
  
 **for** strip **in** self.strips.keys(): *# running on all strips* strip\_name = strip  
 starting\_point = str(self.strips[strip].starting\_point)  
 volume = str(self.strips[strip].volume)  
 file\_path = **f'groups\_files\\{**self.group\_name**}\\{**strip\_name**}.wav'** *# file path  
  
 # sending strip name* self.file\_servers[ip].send([ip], ServerProtocol.strip\_name(strip\_name))  
 *# sending starting point* self.file\_servers[ip].send([ip], ServerProtocol.send\_start\_time(starting\_point))  
 *# sending volume* self.file\_servers[ip].send([ip], ServerProtocol.send\_volume(volume))  
 *# sending file* self.file\_servers[ip].send\_file(ip, file\_path)  
  
 **def** start\_edit\_strip(self, ip, strip\_name):  
 *"""  
 sending to clients strip is being edit* **:param** *ip: ip* **:param** *strip\_name: strip name* **:return***: None  
 """* username = self.usernames[ip]  
 **if** strip\_name **in** self.strip\_edit.keys():  
 *# sending to client strip is occupied* self.comm.send([ip], ServerProtocol.strip\_occupied(self.strip\_edit[strip\_name]))  
 **else**:  
 *# sending to client that he can edit the strip* self.comm.send([ip], **'34'**)  
 self.strip\_edit[strip\_name] = username  
 *# sending to all clients* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 *# sending message* self.comm.send([i], ServerProtocol.start\_edit\_strip(strip\_name, username))  
  
 **def** stop\_edit\_strip(self, ip, strip\_name):  
 *"""  
 sending to clients strip is not being edit* **:param** *ip: ip* **:param** *strip\_name: strip name* **:return***: None  
 """* **del** self.strip\_edit[strip\_name]  
 *# sending to all clients* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 *# sending message* self.comm.send([i], ServerProtocol.stop\_edit\_strip(strip\_name))  
  
 **def** remove\_client\_editing(self, ip):  
 *"""  
 removing client from self.strip\_edit* **:param** *ip: client ip* **:return***: None  
 """* username = self.usernames[ip]  
 **if** username **in** self.strip\_edit.values():  
 **for** strip **in** self.strip\_edit.keys():  
 **if** username == self.strip\_edit[strip]:  
 self.stop\_edit\_strip(ip, strip)  
 **break  
  
 def** cut\_beginning(self, ip, strip\_name, cut\_time):  
 *"""  
 cutting from beginning* **:param** *ip: client ip* **:param** *strip\_name: strip name* **:param** *cut\_time: cut time* **:return***: None  
 """  
 # cutting strip* self.strips[strip\_name].trim\_start(float(cut\_time))  
 *# saving file* self.strips[strip\_name].save\_file()  
 *# sending to clients to cut their strip* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 self.comm.send([i], ServerProtocol.shorten\_start(strip\_name, cut\_time))  
  
 **def** cut\_end(self, ip, strip\_name, cut\_time):  
 *"""  
 cutting from end* **:param** *ip: client ip* **:param** *strip\_name: strip name* **:param** *cut\_time: cut time* **:return***: None  
 """  
 # cutting strip* self.strips[strip\_name].trim\_end(float(cut\_time))  
 *# saving file* self.strips[strip\_name].save\_file()  
 *# sending to clients to cut their strip* **for** i **in** self.comm.open\_clients.values():  
 **if** i != ip:  
 self.comm.send([i], ServerProtocol.shorten\_end(strip\_name, cut\_time))  
  
 **def** check\_strip\_exists(self, ip, strip\_name):  
 *"""  
 checking if strip exists and telling client* **:param** *strip\_name: strip name* **:param** *ip: client ip* **:return***: None  
 """* **if** strip\_name **in** self.strips.keys():  
 self.comm.send([ip], **'31'**)  
 **else**:  
 self.comm.send([ip], **'30'**)  
 *# adding name to strips* self.strips[strip\_name] = **None  
  
 def** set\_strips(self):  
 *"""  
 setting strips* **:return***: None  
 """* db = DB() *# data base object* strips = db.get\_details(self.group\_name.split(**';'**)[0], self.group\_name.split(**';'**)[1])  
 **if** strips **is not None**:  
 **for** strip **in** strips.split(**'|'**):  
 self.strips[strip.split(**';'**)[0]] = MusicStrip(**None**, strip.split(**';'**)[0], float(strip.split(**';'**)[1])  
 , int(strip.split(**';'**)[2]), **f'groups\_files\\{**self.group\_name**}'**)  
  
 **def** save\_file(self):  
 *"""  
 saving edit file in data base* **:return***: None  
 """* db = DB() *# data base object* details = **''** *# details to save* counter = 0 *# counter* **for** strip **in** self.strips.keys():  
 *# writing details* **if** counter > 0:  
 details += **'|'** details += **f'{**strip**};{**self.strips[strip].starting\_point**};{**self.strips[strip].volume**}'** counter += 1  
 *# saving details in data base* db.update\_file(self.group\_name.split(**';'**)[0], self.group\_name.split(**';'**)[1], details)  
 *# deleting all files from undo folder* path = **f'groups\_files\\{**self.group\_name**}\\undo'  
 for** f **in** os.listdir(path):  
 os.remove(os.path.join(path, f))  
 *# closing data base* db.close\_db()

## edit\_group\_com.py

**import** socket  
**import** threading  
**import** select  
**from** server\_protocol **import** ServerProtocol  
  
  
**class** EditGroupCom:  
 *"""  
 edit group server communication  
 """* **def** \_\_init\_\_(self, msg\_q, encrypt, socket, ip):  
 *"""  
 initializing object parameters and calling the thread main loop* **:param** *ip: client ip* **:param** *msg\_q: queue for messages* **:param** *encrypt: encrypt decrypt object  
 """* self.msg\_q = msg\_q  
 self.encrypt = encrypt  
 self.open\_clients = {socket: ip} *# dictionary of open clients* self.running = **True** *# calling main\_loop thread* threading.Thread(target=self.\_main\_loop).start()  
  
 **def** \_main\_loop(self):  
 *"""  
 creating server socket starts to accept new clients  
 putting their messages in msg\_q and calling \_diffie() for new clients* **:return***: None  
 """* **while** self.running:  
 **if** len(self.open\_clients.keys()) > 0:  
 rlist, wlist, xlist = select.select(list(self.open\_clients.keys()), [], [], 0.1)  
 **for** current\_socket **in** rlist:  
  
 **if** current\_socket **in** self.open\_clients.keys():  
 **try**:  
 data\_len = current\_socket.recv(5).decode() *# getting data length* data = current\_socket.recv(int(data\_len)) *# getting data  
 # decrypting data* data = self.encrypt.decrypt(data, self.open\_clients[current\_socket]).decode()  
 **except** Exception **as** e:  
 print(**"EditGroupCom - \_main\_loop"** + str(e))  
 self.\_disconnect\_client(current\_socket)  
 **else**:  
 **if** data == **""**:  
 self.\_disconnect\_client(current\_socket)  
 **else**:  
 **if** current\_socket **in** self.open\_clients.keys(): *# if socket open clients* **if** type(data) == str: *# if not receive file* data = ServerProtocol.unpack(data)  
 self.msg\_q.put((self.open\_clients[current\_socket], data))  
 **else**:  
 **break  
  
 def** send(self, ips, msg):  
 *"""  
 sends msg to list of sockets* **:param** *ips: list of ips* **:param** *msg: message to send* **:return***: None  
 """  
  
 # sending message to all clients* **for** ip **in** ips:  
 soc = self.\_find\_socket\_by\_ip(ip)  
  
 **try**:  
 enc = self.encrypt.encrypt(msg, self.open\_clients[soc])  
 *# sending length of message* soc.send(str(len(enc)).zfill(5).encode())  
 *# sending message* soc.send(enc)  
 **except** Exception **as** e:  
 print(**'EditGroupCom send - '**, e)  
 self.\_disconnect\_client(soc)  
  
 **def** \_disconnect\_client(self, socket):  
 *"""  
 disconnects socket and removes socket from open\_clients* **:param** *socket: socket to disconnect* **:return***: None  
 """  
 # telling server to delete username* self.msg\_q.put((self.open\_clients[socket], [0]))  
 **if** len(self.open\_clients.keys()) == 1:  
 self.msg\_q.put((self.open\_clients[socket], [-1]))  
 self.running = **False** *# removing socket* **del** self.open\_clients[socket]  
 socket.close() *# closing socket* **def** \_find\_socket\_by\_ip(self, ip):  
 *"""  
 returning socket from ip* **:param** *ip: ip* **:return***: socket  
 """* socket = **None  
 for** soc **in** self.open\_clients.keys():  
 **if** self.open\_clients[soc] == ip:  
 socket = soc  
 **break  
 return** socket

## encrypt.py

**from** Cryptodome.Cipher **import** AES  
**from** Cryptodome.Hash **import** SHA256  
**import** base64  
**import** hashlib  
**from** Cryptodome **import** Random  
**from** Cryptodome.Util.Padding **import** unpad  
  
  
**class** Encrypt:  
 *"""  
 class to encrypt and decrypt messages  
 """* **def** \_\_init\_\_(self):  
 *"""  
 initializing keys dictionary and creating Encrypt object  
 """* self.encrypt\_keys = {} *# dictionary of clients keys* self.bs = AES.block\_size *# block size* **def** add\_key(self, key, ip=**None**):  
 *"""  
 adds key to encrypt\_keys* **:param** *key: key of client* **:param** *ip: ip of client* **:return***: None  
 """* self.encrypt\_keys[ip] = key  
  
 **def** encrypt(self, raw, ip=**None**):  
 *"""  
 encrypting raw with client key* **:param** *raw: data to encrypt* **:param** *ip: ip of client to get key* **:return***: returns the encrypted message  
 """* **if** ip:  
 key = hashlib.sha256(self.encrypt\_keys[ip].encode()).digest()  
 **else**:  
 key = hashlib.sha256(list(self.encrypt\_keys.values())[0].encode()).digest()  
  
 *# change to bytes* **if** type(raw) == str:  
 raw = raw.encode()  
  
 raw = self.\_pad(raw)  
  
 iv = Random.new().read(self.bs)  
 cipher = AES.new(key, AES.MODE\_CBC, iv)  
 **return** base64.b64encode(iv + cipher.encrypt(raw))  
  
 **def** decrypt(self, enc, ip=**None**):  
 *"""  
 decrypting raw with client key* **:param** *enc: encrypted data* **:param** *ip: ip of client to get key* **:return***: returns the decrypted message  
 """* **if** ip:  
 key = hashlib.sha256(self.encrypt\_keys[ip].encode()).digest()  
 **else**:  
 key = hashlib.sha256(list(self.encrypt\_keys.values())[0].encode()).digest()  
  
 enc = base64.b64decode(enc)  
 iv = enc[:self.bs]  
 cipher = AES.new(key, AES.MODE\_CBC, iv)  
 **return** unpad(cipher.decrypt(enc[self.bs:]), self.bs)  
  
 **def** \_pad(self, s):  
 *"""  
 Internal method, pads bytes for encryption* **:param** *s: Bytes to pad* **:return***: Padded bytes  
 """* s = (s + (self.bs - len(s) % self.bs) \* bytes([self.bs - len(s) % self.bs]))  
 **return** s  
  
 @staticmethod  
 **def** hash(message):  
 h = SHA256.new()  
 h.update(message.encode())  
 **return** h.hexdigest()  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
  
 **pass**

## gui.py

**import** wx  
**from** client\_logic **import** \*  
**from** pubsub **import** pub  
**from** client\_protocol **import** ClientProtocol  
**import** wx.lib.scrolledpanel  
**from** wx.adv **import** Animation, AnimationCtrl  
**import** wx.lib.intctrl  
  
  
**class** LogIn(wx.Panel):  
 *"""login panel"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent=parent)  
  
 box = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* btn\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# sizer for buttons* username\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# sizer for username* password\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# sizer for password* box.AddSpacer(30)  
  
 self.SetBackgroundColour(**'#D14081'**)  
 *# title* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* txt\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font  
  
 # logo* png = wx.Image(**'logo.png'**, wx.BITMAP\_TYPE\_ANY).ConvertToBitmap() *# logo image* self.bitmap = wx.StaticBitmap(self, -1, png, (png.GetWidth(), png.GetHeight()))  
 box.Add(self.bitmap, 0, wx.ALIGN\_CENTER)  
 box.AddSpacer(80)  
  
 *# enter password and username* self.username\_txt = wx.StaticText(self, -1, **"username: "**) *# username* self.username\_txt.SetFont(btn\_font)  
 self.password\_txt = wx.StaticText(self, -1, **"password: "**) *# password* self.password\_txt.SetFont(btn\_font)  
  
 self.username = wx.TextCtrl(self, size=(200, 30))  
 self.username.SetFont(txt\_font)  
 self.password = wx.TextCtrl(self, size=(200, 30), style=wx.TE\_PASSWORD)  
 self.password.SetFont(txt\_font)  
  
 *# adding username* username\_sizer.Add(self.username\_txt, 0)  
 username\_sizer.AddSpacer(10)  
 username\_sizer.Add(self.username, 0)  
 box.Add(username\_sizer, 0, wx.ALIGN\_CENTER)  
  
 box.AddSpacer(150)  
  
 *# adding password* password\_sizer.Add(self.password\_txt, 0)  
 password\_sizer.AddSpacer(10)  
 password\_sizer.Add(self.password, 0)  
 box.Add(password\_sizer, 0, wx.ALIGN\_CENTER)  
  
 box.AddSpacer(100)  
  
 *# creating buttons* self.log\_in = wx.Button(self, -1, **"log in"**) *# log in btn* self.sign\_in = wx.Button(self, -1, **"sign in"**) *# sign in btn* self.log\_in.SetSize((100, 50))  
 self.sign\_in.SetSize((100, 50))  
  
 *# buttons settings* self.log\_in.SetFont(btn\_font)  
 self.sign\_in.SetFont(btn\_font)  
 self.log\_in.SetBackgroundColour((255, 255, 255, 255))  
 self.sign\_in.SetBackgroundColour((255, 255, 255, 255))  
  
 *# adding btns* btn\_sizer.Add(self.log\_in, 0)  
 btn\_sizer.AddSpacer(200)  
 btn\_sizer.Add(self.sign\_in, 0)  
 box.Add(btn\_sizer, 0, wx.ALIGN\_CENTER)  
  
 *# alignments* self.SetSizer(box)  
  
  
**class** SignIn(wx.Panel):  
 *"""sign in panel"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent)  
  
 box = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* btn\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# sizer for buttons* username\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# sizer for username* password\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# sizer for password* box.AddSpacer(50)  
  
 self.SetBackgroundColour(**'#D14081'**)  
 *# title* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* txt\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font  
  
 # logo* png = wx.Image(**'logo.png'**, wx.BITMAP\_TYPE\_ANY).ConvertToBitmap() *# logo image* self.bitmap = wx.StaticBitmap(self, -1, png, (png.GetWidth(), png.GetHeight()))  
 box.Add(self.bitmap, 0, wx.ALIGN\_CENTER)  
 box.AddSpacer(80)  
  
 *# enter password and username* self.username\_txt = wx.StaticText(self, -1, **"username: "**) *# username* self.username\_txt.SetFont(btn\_font)  
 self.password\_txt = wx.StaticText(self, -1, **"password: "**) *# password* self.password\_txt.SetFont(btn\_font)  
  
 self.username = wx.TextCtrl(self, size=(200, 30))  
 self.username.SetFont(txt\_font)  
 self.password = wx.TextCtrl(self, size=(200, 30), style=wx.TE\_PASSWORD)  
 self.password.SetFont(txt\_font)  
  
 *# adding username* username\_sizer.Add(self.username\_txt, 0)  
 username\_sizer.AddSpacer(10)  
 username\_sizer.Add(self.username, 0)  
 box.Add(username\_sizer, 0, wx.ALIGN\_CENTER)  
  
 box.AddSpacer(150)  
  
 *# adding password* password\_sizer.Add(self.password\_txt, 0)  
 password\_sizer.AddSpacer(10)  
 password\_sizer.Add(self.password, 0)  
 box.Add(password\_sizer, 0, wx.ALIGN\_CENTER)  
  
 box.AddSpacer(100)  
  
 *# creating buttons* self.back = wx.Button(self, -1, **"back"**) *# log in btn* self.sign\_in = wx.Button(self, -1, **"sign in"**) *# sign in btn* self.back.SetSize((100, 50))  
 self.sign\_in.SetSize((100, 50))  
  
 *# buttons settings* self.back.SetFont(btn\_font)  
 self.back.SetForegroundColour(wx.Colour(255, 255, 255))  
 self.sign\_in.SetFont(btn\_font)  
 self.back.SetBackgroundColour((0, 0, 0, 0))  
 self.sign\_in.SetBackgroundColour((255, 255, 255, 255))  
  
 *# adding btns* btn\_sizer.Add(self.back, 0)  
 btn\_sizer.AddSpacer(200)  
 btn\_sizer.Add(self.sign\_in, 0)  
 box.Add(btn\_sizer, 0, wx.ALIGN\_CENTER)  
  
 *# alignments* self.SetSizer(box)  
  
  
**class** Home(wx.Panel):  
 *"""sign in panel"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent)  
 self.edit\_files = [] *# list of edit files* box = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* box.AddSpacer(50)  
  
 *# logo* png = wx.Image(**'logo.png'**, wx.BITMAP\_TYPE\_ANY).ConvertToBitmap() *# logo image* self.bitmap = wx.StaticBitmap(self, -1, png, (png.GetWidth(), png.GetHeight()))  
 box.Add(self.bitmap, 0, wx.ALIGN\_CENTER)  
  
 self.SetBackgroundColour(**'#D14081'**)  
 *# title* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* txt\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font  
  
 # choose file text* cblbl = wx.StaticText(self, label=**"choose file:"**, style=wx.ALIGN\_CENTRE)  
 cblbl.SetFont(txt\_font)  
 box.Add(cblbl, 0, wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALL)  
  
 *# combo to select files* self.choice\_box = wx.Choice(self, choices=self.edit\_files, size=(610, 50))  
 font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font* self.choice\_box.SetFont(font)  
 box.AddSpacer(10)  
 box.Add(self.choice\_box, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 btn\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 *# buttons* self.edit\_file = wx.Button(self, -1, **"edit file"**) *# edit file btn* self.add\_file = wx.Button(self, -1, **"add file"**) *# add file in btn* self.edit\_file.SetSize((100, 50))  
 self.add\_file.SetSize((100, 50))  
  
 *# buttons settings* self.edit\_file.SetFont(btn\_font)  
 self.add\_file.SetFont(btn\_font)  
 self.edit\_file.SetBackgroundColour((255, 255, 255, 255))  
 self.add\_file.SetBackgroundColour((255, 255, 255, 255))  
  
 *# adding btns* btn\_sizer.Add(self.edit\_file, 0)  
 btn\_sizer.AddSpacer(200)  
 btn\_sizer.Add(self.add\_file, 0)  
 box.AddSpacer(300)  
  
 box.Add(btn\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 *# alignments* self.SetSizer(box)  
  
  
**class** AddFile(wx.Panel):  
 *"""add file panel"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent)  
 self.usernames = [] *# list of users* box = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* box.AddSpacer(50)  
  
 *# logo* png = wx.Image(**'logo.png'**, wx.BITMAP\_TYPE\_ANY).ConvertToBitmap() *# logo image* self.bitmap = wx.StaticBitmap(self, -1, png, (png.GetWidth(), png.GetHeight()))  
 box.Add(self.bitmap, 0, wx.ALIGN\_CENTER)  
  
 self.SetBackgroundColour(**'#D14081'**)  
 *# title* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* txt\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font  
  
 # choose file text* cblbl = wx.StaticText(self, label=**"select users to edit with"**, style=wx.ALIGN\_CENTRE)  
 cblbl.SetFont(txt\_font)  
 box.Add(cblbl, 0, wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALL)  
  
 *# check list box to select users* self.choice\_box = wx.CheckListBox(self, -1, choices=self.usernames, size=(610, 100))  
 font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font* self.choice\_box.SetFont(font)  
 box.AddSpacer(10)  
 box.Add(self.choice\_box, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 box.AddSpacer(100)  
  
 *# text ctrl to fill file name* file\_txt = wx.StaticText(self, label=**"enter file name"**, style=wx.ALIGN\_CENTRE)  
 file\_txt.SetFont(txt\_font)  
 self.file\_name = wx.TextCtrl(self, size=(400, 30))  
 self.file\_name.SetFont(txt\_font)  
 box.Add(file\_txt, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 box.Add(self.file\_name, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 btn\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 *# buttons* self.create\_file = wx.Button(self, -1, **"create file"**) *# edit file btn* self.back = wx.Button(self, -1, **"back"**) *# add file in btn* self.create\_file.SetSize((100, 50))  
 self.back.SetSize((100, 50))  
  
 *# buttons settings* self.create\_file.SetFont(btn\_font)  
 self.back.SetFont(btn\_font)  
 self.back.SetForegroundColour(wx.Colour(255, 255, 255))  
 self.create\_file.SetBackgroundColour((255, 255, 255, 255))  
 self.back.SetBackgroundColour((0, 0, 0, 0))  
  
 *# adding btns* btn\_sizer.Add(self.create\_file, 0)  
 btn\_sizer.AddSpacer(200)  
 btn\_sizer.Add(self.back, 0)  
 box.AddSpacer(100)  
  
 *# adding btns* box.Add(btn\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 *# alignments* self.SetSizer(box)  
  
  
**class** EditFile(wx.Panel):  
 *"""edit file panel"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent)  
  
 self.usernames = [] *# list of connected users  
 # strips dictionary = {strip name : list of widgets}* self.strips = {}  
 *# {strip name : [is\_editing, username}* self.strip\_edit = {}  
  
 *# vertical sizer* box = wx.BoxSizer(wx.VERTICAL)  
  
 *# screen width and height for sizing* screenWidth, screenHeight = wx.DisplaySize()  
  
 self.SetBackgroundColour(**'#D14081'**) *# setting background color* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* txt\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font* txt2\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL, underline=**True**) *# font  
  
 # creating buttons* self.add\_strip = wx.Button(self, label=**"add strip"**, size=(150, 50))  
 self.add\_strip.SetBackgroundColour(**'#40d190'**)  
 self.add\_strip.SetFont(btn\_font)  
 self.back = wx.Button(self, label=**"back"**, size=(100, 50)) *# back btn* self.back.SetBackgroundColour((0, 0, 0, 0))  
 self.back.SetFont(btn\_font)  
 self.back.SetForegroundColour(wx.Colour(255, 255, 255))  
 self.play\_audio = wx.Button(self, label=**"play audio"**, size=(150, 50))  
 self.play\_audio.SetBackgroundColour(**'#40d190'**)  
 self.play\_audio.SetFont(btn\_font)  
 self.save\_file = wx.Button(self, label=**"save file"**, size=(150, 50))  
 self.save\_file.SetBackgroundColour(**'#40d190'**)  
 self.save\_file.SetFont(btn\_font)  
  
 *# edit panel* hoz\_edit\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# horizontal sizer* self.edit\_panel = wx.Panel(self, size=(int(screenWidth\*0.35), int(screenHeight\*0.75)), style=wx.NO\_BORDER)  
 self.edit\_panel.SetBackgroundColour(**'#5A5A5A'**) *# setting background color* hoz\_edit\_sizer.AddSpacer(20)  
 hoz\_edit\_sizer.Add(self.edit\_panel, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 *# real edit panel* hoz\_real\_edit\_sizer = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* self.edit\_edit\_panel = wx.Panel(self.edit\_panel, size=(int(screenWidth \* 0.3)  
 , int(screenHeight \* 0.5625)), style=wx.NO\_BORDER)  
 self.edit\_edit\_panel.SetBackgroundColour(**'#5A5A5A'**) *# setting background color  
  
 # edit strip panel widgets* volume = wx.StaticText(self.edit\_edit\_panel, label=**"volume:"**) *# volume* volume.SetFont(txt2\_font)  
 self.volume = wx.StaticText(self.edit\_edit\_panel, label=**"50%"**) *# volume* self.volume.SetFont(txt\_font)  
 self.volume.SetBackgroundColour(**'#FFFFFF'**) *# volume background color  
 # volume butons* plus = wx.Bitmap(**'plus.png'**) *# create wx.Bitmap object* minus = wx.Bitmap(**'minus.png'**) *# create wx.Bitmap object* self.plus = wx.Button(self.edit\_edit\_panel, size=(30, 30), style=wx.NO\_BORDER)  
 self.plus.SetBackgroundColour(**'#5A5A5A'**)  
 self.plus.SetBitmap(plus) *# set bmp as bitmap for button* self.minus = wx.Button(self.edit\_edit\_panel, size=(30, 30), style=wx.NO\_BORDER)  
 self.minus.SetBackgroundColour(**'#5A5A5A'**)  
 self.minus.SetBitmap(minus) *# set bmp as bitmap for button  
 # volume sizer* volume\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# horizontal sizer* volume\_sizer.Add(self.minus, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 volume\_sizer.AddSpacer(5)  
 volume\_sizer.Add(self.volume, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 volume\_sizer.AddSpacer(5)  
 volume\_sizer.Add(self.plus, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_real\_edit\_sizer.AddSpacer(10)  
 hoz\_real\_edit\_sizer.Add(volume, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_real\_edit\_sizer.Add(volume\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 *# cutting buttons and ctrls* mili = wx.StaticText(self.edit\_edit\_panel, label=**"cut milliseconds from strip:"**) *# cut milliseconds* mili.SetFont(txt2\_font)  
 self.beginning = wx.lib.intctrl.IntCtrl(self.edit\_edit\_panel, size=(100, 30))  
 self.beginning.SetFont(txt\_font)  
 self.end = wx.lib.intctrl.IntCtrl(self.edit\_edit\_panel, size=(100, 30))  
 self.end.SetFont(txt\_font)  
 cut\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# horizontal sizer* cut\_sizer.Add(self.beginning, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 cut\_sizer.AddSpacer(150)  
 cut\_sizer.Add(self.end, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_real\_edit\_sizer.AddSpacer(20)  
 hoz\_real\_edit\_sizer.Add(mili, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_real\_edit\_sizer.AddSpacer(10)  
 hoz\_real\_edit\_sizer.Add(cut\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 *# adding cutting buttons* self.cut\_beginning = wx.Button(self.edit\_edit\_panel, size=(150, 30), style=wx.NO\_BORDER, label=**'cut beginning'**)  
 self.cut\_beginning.SetFont(btn\_font)  
 self.cut\_beginning.SetBackgroundColour(**'#40d190'**)  
 self.cut\_end = wx.Button(self.edit\_edit\_panel, size=(150, 30), style=wx.NO\_BORDER, label=**'cut end'**)  
 self.cut\_end.SetFont(btn\_font)  
 self.cut\_end.SetBackgroundColour(**'#40d190'**)  
 cut\_button\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# horizontal sizer* cut\_button\_sizer.Add(self.cut\_beginning, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 cut\_button\_sizer.AddSpacer(100)  
 cut\_button\_sizer.Add(self.cut\_end, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_real\_edit\_sizer.AddSpacer(10)  
 hoz\_real\_edit\_sizer.Add(cut\_button\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 *# seting strip edit panel sizer* self.edit\_edit\_panel.SetSizer(hoz\_real\_edit\_sizer)  
  
 *# scroller* self.scroll\_vert = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* hoz\_scroll\_sizer = wx.BoxSizer(wx.HORIZONTAL) *# vertical sizer* self.strips\_scroller = wx.lib.scrolledpanel.ScrolledPanel(self, -1, style=wx.TAB\_TRAVERSAL | wx.SUNKEN\_BORDER,  
 size=(int(screenWidth\*0.6), int(screenHeight\*0.75)))  
 self.strips\_scroller.SetBackgroundColour(**'#5A5A5A'**)  
 self.strips\_scroller.SetAutoLayout(1)  
 self.strips\_scroller.SetupScrolling()  
 self.strips\_scroller.SetSizer(self.scroll\_vert)  
  
 *# edit panel widgets* vert\_edit\_sizer = wx.BoxSizer(wx.VERTICAL) *# sizer for edit panel vertical* self.strip\_name = wx.StaticText(self.edit\_panel, label=**" "**) *# strip name* self.strip\_name.SetFont(txt2\_font)  
 self.strip\_length = wx.StaticText(self.edit\_panel, label=**" "**) *# length of strip* self.strip\_length.SetFont(txt\_font)  
 self.strip\_length.SetForegroundColour(wx.Colour(255, 255, 255))  
 vert\_edit\_sizer.AddSpacer(20)  
 *# adding strip length and name to sizer* vert\_edit\_sizer.Add(self.strip\_name, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 vert\_edit\_sizer.Add(self.strip\_length, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 *# button to mute* self.mute = wx.Button(self.edit\_panel, label=**"mute"**, size=(70, 50))  
 self.mute.SetBackgroundColour(**'#40d190'**)  
 self.mute.SetFont(btn\_font)  
 *# button to edit strip* self.edit = wx.Button(self.edit\_panel, label=**"edit strip"**, size=(120, 50))  
 self.edit.SetBackgroundColour(**'#40d190'**)  
 self.edit.SetFont(btn\_font)  
 edit\_button\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 edit\_button\_sizer.Add(self.mute, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 edit\_button\_sizer.AddSpacer(20)  
 edit\_button\_sizer.Add(self.edit, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 vert\_edit\_sizer.AddSpacer(20)  
 vert\_edit\_sizer.Add(edit\_button\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 *# adding real edit panel* vert\_edit\_sizer.Add(self.edit\_edit\_panel, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 self.edit\_edit\_panel.Hide()  
  
 *# scroll horizontal sizer* hoz\_scroll\_sizer.Add(self.edit\_panel, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_scroll\_sizer.AddSpacer(20)  
 hoz\_scroll\_sizer.Add(self.strips\_scroller, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 hoz\_scroll\_sizer.AddSpacer(20)  
  
 *# adding button* btn\_sizer\_top = wx.BoxSizer(wx.HORIZONTAL)  
  
 btn\_sizer\_top.Add(self.add\_strip, 0, wx.RIGHT | wx.ALIGN\_RIGHT, int(screenWidth\*0.6)-250)  
 btn\_sizer\_top.Add(self.back, 0, wx.RIGHT | wx.ALIGN\_RIGHT, 20)  
 box.Add(btn\_sizer\_top, 0, wx.ALIGN\_RIGHT)  
  
 *# adding scroll* box.AddSpacer(20)  
 box.Add(hoz\_scroll\_sizer, 0, wx.ALIGN\_RIGHT)  
  
 *# adding bottom buttons* btn\_sizer\_bottom = wx.BoxSizer(wx.HORIZONTAL)  
 btn\_sizer\_bottom.Add(self.play\_audio, 0, wx.RIGHT | wx.ALIGN\_RIGHT, int(screenWidth \* 0.6) - 300)  
 btn\_sizer\_bottom.Add(self.save\_file, 0, wx.RIGHT | wx.ALIGN\_RIGHT, 20)  
 box.AddSpacer(20)  
 box.Add(btn\_sizer\_bottom, 0, wx.ALIGN\_RIGHT)  
  
 *# alignments* self.SetSizer(box)  
 self.edit\_panel.SetSizer(vert\_edit\_sizer)  
  
 *# hiding edit panel* self.edit\_panel.Hide()  
  
  
**class** AddStrip(wx.Panel):  
 *"""add file strip"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent)  
 self.strip\_file = **''** *# strip file to upload* self.strip = **''** *# strip name* box = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* box.AddSpacer(50)  
  
 *# logo* png = wx.Image(**'logo.png'**, wx.BITMAP\_TYPE\_ANY).ConvertToBitmap() *# logo image* self.bitmap = wx.StaticBitmap(self, -1, png, (png.GetWidth(), png.GetHeight()))  
 box.Add(self.bitmap, 0, wx.ALIGN\_CENTER)  
 self.SetBackgroundColour(**'#D14081'**)  
  
 *# font* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* txt\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_NORMAL) *# font  
  
 # text ctrl to fill strip name* file\_txt = wx.StaticText(self, label=**"enter strip name"**, style=wx.ALIGN\_CENTRE)  
 file\_txt.SetFont(txt\_font)  
 self.strip\_name = wx.TextCtrl(self, size=(400, 30))  
 self.strip\_name.SetFont(txt\_font)  
 box.Add(file\_txt, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
 box.Add(self.strip\_name, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 *# select file button* self.select\_file = wx.Button(self, -1, **"choose a file"**)  
 box.Add(self.select\_file, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 btn\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
  
 *# buttons* self.create\_strip = wx.Button(self, -1, **"add strip"**) *# edit file btn* self.back = wx.Button(self, -1, **"back"**) *# add file in btn* self.create\_strip.SetSize((100, 50))  
 self.back.SetSize((100, 50))  
  
 *# buttons settings* self.create\_strip.SetFont(btn\_font)  
 self.back.SetFont(btn\_font)  
 self.back.SetForegroundColour(wx.Colour(255, 255, 255))  
 self.create\_strip.SetBackgroundColour((255, 255, 255, 255))  
 self.back.SetBackgroundColour((0, 0, 0, 0))  
  
 *# adding btns* btn\_sizer.Add(self.create\_strip, 0)  
 btn\_sizer.AddSpacer(200)  
 btn\_sizer.Add(self.back, 0)  
 box.AddSpacer(100)  
  
 *# adding btns* box.Add(btn\_sizer, 0, wx.ALIGN\_CENTER\_HORIZONTAL)  
  
 *# alignments* self.SetSizer(box)  
  
  
**class** Loading(wx.Panel):  
 *"""add file strip"""* **def** \_\_init\_\_(self, parent):  
 wx.Panel.\_\_init\_\_(self, parent)  
 box = wx.BoxSizer(wx.VERTICAL) *# vertical sizer* self.SetBackgroundColour(**'#D14081'**)  
  
 *# showing loading widget* loading = Animation()  
 loading.LoadFile(**'loading.gif'**)  
 ctrl = AnimationCtrl(self, -1, loading)  
 ctrl.Play()  
 box.Add(ctrl, 0, wx.ALIGN\_CENTER\_HORIZONTAL | wx.ALIGN\_CENTER\_VERTICAL)  
 *# alignments* self.SetSizer(box)  
  
  
**class** Program(wx.Frame):  
 *"""graphic frame contains all panels"""* **def** \_\_init\_\_(self, comm):  
 wx.Frame.\_\_init\_\_(self, **None**, wx.ID\_ANY, **'Program'**, style=wx.DEFAULT\_FRAME\_STYLE)  
 *# client comm* self.comm = comm  
 *# file comm* self.file\_comm = **None** *# username of client* self.username = **''** sizer = wx.BoxSizer()  
 self.SetSizer(sizer)  
 self.Maximize(**True**)  
  
 *# panels* self.sign\_in\_panel = SignIn(self)  
 self.log\_in\_panel = LogIn(self)  
 self.home\_panel = Home(self)  
 self.add\_file\_panel = AddFile(self)  
 self.edit\_panel = EditFile(self)  
 self.add\_strip\_panel = AddStrip(self)  
 self.loading\_panel = Loading(self)  
  
 *# log in panel* sizer.Add(self.log\_in\_panel, 1, wx.EXPAND)  
 self.log\_in\_panel.log\_in.Bind(wx.EVT\_BUTTON, self.log\_in)  
 self.log\_in\_panel.sign\_in.Bind(wx.EVT\_BUTTON, self.show\_panel\_two)  
  
 *# sign in panel* sizer.Add(self.sign\_in\_panel, 1, wx.EXPAND)  
 self.sign\_in\_panel.sign\_in.Bind(wx.EVT\_BUTTON, self.sign\_in)  
 self.sign\_in\_panel.back.Bind(wx.EVT\_BUTTON, self.show\_panel\_one)  
  
 *# home panel* sizer.Add(self.home\_panel, 1, wx.EXPAND)  
 self.home\_panel.edit\_file.Bind(wx.EVT\_BUTTON, self.start\_editing)  
 self.home\_panel.add\_file.Bind(wx.EVT\_BUTTON, self.show\_panel\_four)  
  
 *# add file panel* sizer.Add(self.add\_file\_panel, 1, wx.EXPAND)  
 self.add\_file\_panel.back.Bind(wx.EVT\_BUTTON, self.show\_panel\_three)  
 self.add\_file\_panel.create\_file.Bind(wx.EVT\_BUTTON, self.create\_file)  
  
 *# edit file panel* sizer.Add(self.edit\_panel, 1, wx.EXPAND)  
 self.edit\_panel.add\_strip.Bind(wx.EVT\_BUTTON, self.show\_panel\_six)  
 self.edit\_panel.play\_audio.Bind(wx.EVT\_BUTTON, self.play\_audio)  
 self.edit\_panel.mute.Bind(wx.EVT\_BUTTON, self.mute\_unmute)  
 self.edit\_panel.edit.Bind(wx.EVT\_BUTTON, self.start\_edit\_strip)  
 self.edit\_panel.plus.Bind(wx.EVT\_BUTTON, self.inc)  
 self.edit\_panel.minus.Bind(wx.EVT\_BUTTON, self.dec)  
 self.edit\_panel.cut\_beginning.Bind(wx.EVT\_BUTTON, self.check\_cut\_beginning)  
 self.edit\_panel.cut\_end.Bind(wx.EVT\_BUTTON, self.check\_cut\_end)  
  
 *# add strip panel* sizer.Add(self.add\_strip\_panel, 1, wx.EXPAND)  
 self.add\_strip\_panel.create\_strip.Bind(wx.EVT\_BUTTON, self.request\_strip)  
 self.add\_strip\_panel.select\_file.Bind(wx.EVT\_BUTTON, self.on\_choose\_file)  
 self.add\_strip\_panel.back.Bind(wx.EVT\_BUTTON, self.show\_panel\_five)  
  
 *# loading panel* sizer.Add(self.loading\_panel, 1, wx.EXPAND)  
  
 *# hiding panels* self.sign\_in\_panel.Hide()  
 self.home\_panel.Hide()  
 self.add\_file\_panel.Hide()  
 self.edit\_panel.Hide()  
 self.add\_strip\_panel.Hide()  
 self.loading\_panel.Hide()  
  
 *# pubsubs* pub.subscribe(self.show\_panel\_three, **"success\_login"**)  
 pub.subscribe(self.pop\_error, **"fail\_login"**)  
 pub.subscribe(self.show\_panel\_three, **"success\_sign\_in"**)  
 pub.subscribe(self.pop\_error, **"fail\_sign\_in"**)  
 pub.subscribe(self.set\_users, **"set\_users"**)  
 pub.subscribe(self.pop\_error, **"user\_already\_connected"**)  
 pub.subscribe(self.set\_files, **"set\_files"**)  
 pub.subscribe(self.show\_panel\_five, **"open\_edit"**)  
 pub.subscribe(self.pop\_error, **"file\_exists"**)  
 pub.subscribe(self.add\_user, **"add\_user"**)  
 pub.subscribe(self.add\_file, **"add\_file"**)  
 pub.subscribe(self.set\_file\_comm, **"file\_comm"**)  
 pub.subscribe(self.add\_strip, **"add\_strip"**)  
 pub.subscribe(self.show\_panel\_loading, **"loading"**)  
 pub.subscribe(self.play\_audio, **"music\_stopped"**)  
 pub.subscribe(self.editor\_edit, **"start\_stop\_edit"**)  
 pub.subscribe(self.pop\_error, **"strip\_exists"**)  
 pub.subscribe(self.send\_strip, **"send\_strip"**)  
 pub.subscribe(self.show\_edit\_strip, **"edit\_strip"**)  
 pub.subscribe(self.pop\_error, **"strip\_occupied"**)  
 pub.subscribe(self.inc, **"inc\_volume"**)  
 pub.subscribe(self.dec, **"dec\_volume"**)  
 pub.subscribe(self.cut\_beginning, **"cut\_beginning"**)  
 pub.subscribe(self.cut\_end, **"cut\_end"**)  
  
 **def** log\_in(self, event):  
 *"""  
 checking if password and username are not empty* **:return***: None  
 """* password = self.log\_in\_panel.password.GetValue()  
 username = self.log\_in\_panel.username.GetValue()  
 **if** len(password) > 0 **and** len(username) > 0:  
 *# sending log in message* self.comm.send(ClientProtocol.log\_in(username, password))  
 **else**:  
 self.pop\_error(**"you have to fill in username and password"**)  
  
 event.Skip()  
  
 **def** sign\_in(self, event):  
 *"""  
 checking if password and username are not empty* **:return***: None  
 """* password = self.sign\_in\_panel.password.GetValue()  
 username = self.sign\_in\_panel.username.GetValue()  
  
 *# checking username requirements* **if** len(password) == 0 **or** len(username) == 0:  
 self.pop\_error(**"you have to fill in username and password"**)  
 **elif ',' in** username **or ';' in** username **or '|' in** username:  
 self.pop\_error(**"username cannot contain ',' or ';' or '|'"**)  
 **elif** len(username) > 20:  
 self.pop\_error(**"username can't be longer than 20 characters"**)  
 **elif ' ' in** username:  
 self.pop\_error(**"username can't contain whitespace"**)  
 **else**:  
 *# sending sign in message* self.comm.send(ClientProtocol.sign\_in(username, password))  
 event.Skip()  
  
 **def** start\_editing(self, event):  
 *"""  
 sends start editing message to server* **:return***: None  
 """* index = self.home\_panel.choice\_box.GetSelection()  
 *# checking if list is empty* **if** len(self.home\_panel.edit\_files) == 0:  
 self.pop\_error(**'you have no edit files'**)  
 **elif** index == -1: *# if nothing was selected* self.pop\_error(**'you have to select a file'**)  
 **else**:  
 *# sending to server that we want to edit a file* self.comm.send(ClientProtocol.start\_edit\_file(self.home\_panel.choice\_box.GetString(index)))  
  
 event.Skip()  
  
 **def** create\_file(self, event):  
 *"""  
 checking input and sending create file message to server* **:param** *event: None* **:return***: None  
 """* file\_name = self.add\_file\_panel.file\_name.GetValue()  
  
 *# checking input* **if** len(file\_name) == 0:  
 self.pop\_error(**'you have to fill in file name'**)  
 **elif ',' in** file\_name **or ';' in** file\_name **or '|' in** file\_name:  
 self.pop\_error(**"file\_name cannot contain ',' or ';' or '|'"**)  
 **elif** len(self.add\_file\_panel.choice\_box.GetCheckedStrings()) == 0:  
 self.pop\_error(**"you have to select users to edit with"**)  
 **else**:  
 *# sending create file request to server* file = file\_name + **';'** + self.username + **','** + **','**.join(self.add\_file\_panel.choice\_box.GetCheckedStrings())  
 self.comm.send(ClientProtocol.create\_file(file))  
  
 **def** request\_strip(self, event):  
 *"""  
 checking add strip input and sending add strip to sever* **:param** *event: None* **:return***: None  
 """* strip\_name = self.add\_strip\_panel.strip\_name.GetValue()  
  
 *# checking input* **if** len(strip\_name) == 0:  
 self.pop\_error(**'you have to fill in strip name'**)  
 **elif ',' in** strip\_name **or ';' in** strip\_name **or '|' in** strip\_name:  
 self.pop\_error(**"strip name cannot contain ',' or ';' or '|'"**)  
 **elif** len(self.add\_strip\_panel.strip\_file) == 0:  
 self.pop\_error(**'you have to select strip file'**)  
 **else**:  
 *# sending strip request to server* self.comm.send(ClientProtocol.strip\_name\_request(strip\_name))  
  
 **def** send\_strip(self):  
 *"""  
 sending strip to server* **:return***: None  
 """* strip\_name = self.add\_strip\_panel.strip\_name.GetValue()  
 *# sending strip name* self.file\_comm.send(ClientProtocol.add\_strip(strip\_name))  
 *# sending starting point* self.file\_comm.send(ClientProtocol.send\_start\_time(**'0.0'**))  
 *# sending volume* self.file\_comm.send(ClientProtocol.send\_volume(**'0'**))  
 *# sending file* self.file\_comm.send\_file(self.add\_strip\_panel.strip\_file)  
 *# sending client information about strip* wx.CallAfter(pub.sendMessage, **"add\_strip\_logic"**, file\_path=self.add\_strip\_panel.strip\_file  
 , strip\_name=strip\_name, starting\_point=0.0, volume=0)  
 *# showing edit panel* self.show\_panel\_five(**None**)  
  
 **def** on\_choose\_file(self, event):  
 wildcard = **'Audio files (\*.wav)|\*.wav'** dlg = wx.FileDialog(self, message=**'Choose an audio file'**, wildcard=wildcard,  
 style=wx.FD\_OPEN | wx.FD\_FILE\_MUST\_EXIST)  
 **if** dlg.ShowModal() == wx.ID\_OK:  
 self.add\_strip\_panel.strip\_file = dlg.GetPath() *# strip file path* dlg.Destroy()  
 event.Skip()  
  
 **def** set\_users(self, users):  
 *"""  
 setting users list* **:param** *users: list of users* **:return***: None  
 """  
 # setting client username* self.username = users[0]  
 self.add\_file\_panel.usernames = users[1:]  
 self.add\_file\_panel.choice\_box.Set(self.add\_file\_panel.usernames)  
  
 **def** set\_files(self, file\_names):  
 *"""  
 setting file name* **:param** *file\_names: file names* **:return***: None  
 """* self.home\_panel.choice\_box.Set(file\_names)  
 self.home\_panel.edit\_files = file\_names  
  
 **def** add\_user(self, username):  
 *"""  
 adding user to usernames list* **:param** *username: username to ass* **:return***: None  
 """* self.add\_file\_panel.usernames.append(username)  
 self.add\_file\_panel.choice\_box.Set(self.add\_file\_panel.usernames)  
  
 **def** add\_file(self, file\_name):  
 *"""  
 adding filename to files list* **:param** *file\_name: file name* **:return***: None  
 """* self.home\_panel.edit\_files.append(file\_name)  
 self.home\_panel.choice\_box.Set(self.home\_panel.edit\_files)  
  
 **def** set\_file\_comm(self, file\_comm):  
 *"""  
 setting file comm* **:param** *file\_comm: file comm* **:return***: None  
 """* self.file\_comm = file\_comm  
  
 **def** add\_strip(self, strip\_name, starting\_point, volume, length):  
 *"""  
 adding strip to edit panel* **:param** *strip\_name: strip name* **:param** *starting\_point: starting point* **:param** *volume: volume* **:param** *length: strip length* **:return***: None  
 """  
 # adding strip widget to scroller* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* strip\_btn = wx.Button(self.edit\_panel.strips\_scroller, label=strip\_name, size=(int(length\*50), 80), style=wx.NO\_BORDER)  
 wave = wx.Bitmap(**'wave.png'**) *# create wx.Bitmap object* strip\_btn.SetBitmap(wave)  
 strip\_btn.SetBackgroundColour(**'#483D8B'**)  
 strip\_btn.SetFont(btn\_font)  
 self.edit\_panel.scroll\_vert.AddSpacer(20)  
 strip\_sizer = wx.BoxSizer(wx.HORIZONTAL)  
 strip\_sizer.AddSpacer(int(starting\_point\*50))  
 strip\_sizer.Add(strip\_btn, 0, wx.LEFT | wx.ALIGN\_LEFT)  
 self.edit\_panel.scroll\_vert.Add(strip\_sizer, 0, wx.LEFT | wx.ALIGN\_LEFT)  
 self.edit\_panel.strips\_scroller.SetupScrolling()  
 *# adding widget to strip list* self.edit\_panel.strips[strip\_name] = [strip\_btn, starting\_point, (50+volume), length, **False**, strip\_sizer]  
 *# binding button* strip\_btn.Bind(wx.EVT\_BUTTON, self.show\_edit\_panel)  
  
 **def** show\_edit\_panel(self, event):  
 *""" showing edit panel """* **if** self.edit\_panel.edit.GetLabel() == **'edit strip'**:  
 button = event.GetEventObject()  
 strip = button.GetLabel() *# name of strip* strip\_length = self.edit\_panel.strips[strip][3]  
 muted = self.edit\_panel.strips[strip][4]  
 *# setting widgets* **if** self.edit\_panel.strip\_name.GetLabel() **in** self.edit\_panel.strips.keys(): *# changing color* self.edit\_panel.strips[self.edit\_panel.strip\_name.GetLabel()][0].SetBackgroundColour(**'#483D8B'**)  
 self.edit\_panel.strip\_name.SetLabel(strip)  
 self.edit\_panel.strip\_length.SetLabel(**f'strip length: {**strip\_length\*1000//1/1000**} seconds'**)  
 button.SetBackgroundColour(**'#3D8B48'**) *# changing to green  
 # setting mute button* **if** muted:  
 *# change color of button* self.edit\_panel.mute.SetBackgroundColour(**'#FF0000'**)  
 *# change text* self.edit\_panel.mute.SetLabel(**'unmute'**)  
 **else**:  
 *# change color of button* self.edit\_panel.mute.SetBackgroundColour(**'#40d190'**)  
 *# change text* self.edit\_panel.mute.SetLabel(**'mute'**)  
 self.edit\_panel.edit\_panel.Show()  
 self.edit\_panel.edit\_panel.Layout()  
 self.edit\_panel.Layout()  
 self.Layout()  
 **else**:  
 self.pop\_error(**'you can only edit 1 strip at a time'**)  
  
 **def** play\_audio(self, event):  
 *"""  
 playing audio or stopping audio* **:return***: None  
 """* **if** event:  
 button = event.GetEventObject()  
 **if** button.GetLabel() == **'play audio'**:  
 *# telling logic to play audio* wx.CallAfter(pub.sendMessage, **"play\_music"**)  
 *# change color of button* button.SetBackgroundColour(**'#FF0000'**)  
 *# change text* button.SetLabel(**'stop playing'**)  
 **else**:  
 *# telling logic to play audio* wx.CallAfter(pub.sendMessage, **"stop\_play"**)  
 *# change color of button* button.SetBackgroundColour(**'#40d190'**)  
 *# change text* button.SetLabel(**'play audio'**)  
 event.Skip()  
 **else**:  
 *# change color of button* self.edit\_panel.play\_audio.SetBackgroundColour(**'#40d190'**)  
 *# change text* self.edit\_panel.play\_audio.SetLabel(**'play audio'**)  
 self.Layout()  
  
 **def** mute\_unmute(self, event):  
 *"""  
 changes mute and unmute* **:return***: None  
 """* **if** self.edit\_panel.play\_audio.GetLabel() == **'play audio'**:  
 **if** self.edit\_panel.mute.GetLabel() == **'mute'**:  
 *# telling logic to mute* wx.CallAfter(pub.sendMessage, **"mute"**, strip\_name=self.edit\_panel.strip\_name.GetLabel())  
 *# change color of button* self.edit\_panel.mute.SetBackgroundColour(**'#FF0000'**)  
 *# change text* self.edit\_panel.mute.SetLabel(**'unmute'**)  
 self.edit\_panel.strips[self.edit\_panel.strip\_name.GetLabel()][4] = **True  
 else**:  
 *# telling logic to unmute* wx.CallAfter(pub.sendMessage, **"mute"**, strip\_name=self.edit\_panel.strip\_name.GetLabel())  
 *# change color of button* self.edit\_panel.mute.SetBackgroundColour(**'#40d190'**)  
 *# change text* self.edit\_panel.mute.SetLabel(**'mute'**)  
 self.edit\_panel.strips[self.edit\_panel.strip\_name.GetLabel()][4] = **False** self.Layout()  
 event.Skip()  
  
 **def** start\_edit\_strip(self, event):  
 *"""  
 showing edit panel* **:param** *event: event* **:return***: None  
 """* **if** self.edit\_panel.play\_audio.GetLabel() == **'play audio'**:  
 strip\_name = self.edit\_panel.strip\_name.GetLabel()  
 self.edit\_panel.volume.SetLabel(**f'{**self.edit\_panel.strips[strip\_name][2]**}%'**) *# volume* **if** self.edit\_panel.edit.GetLabel() == **'edit strip'**:  
 *# telling server that we are editing this strip* self.comm.send(ClientProtocol.start\_edit\_strip(strip\_name))  
 **else**:  
 *# sending to server that we want to stop editing* self.comm.send(ClientProtocol.stop\_editing\_strip(strip\_name))  
 *# change color of button* self.edit\_panel.edit.SetBackgroundColour(**'#40d190'**)  
 *# change text* self.edit\_panel.edit.SetLabel(**'edit strip'**)  
 *# hiding edit panel* self.edit\_panel.edit\_edit\_panel.Hide()  
 self.edit\_panel.edit\_panel.Layout()  
 self.edit\_panel.Layout()  
 self.Layout()  
 event.Skip()  
  
 **def** show\_edit\_strip(self):  
 *"""  
 showing edit strip panel* **:return***: None  
 """  
 # change color of button* self.edit\_panel.edit.SetBackgroundColour(**'#FF0000'**)  
 *# change text* self.edit\_panel.edit.SetLabel(**'stop editing'**)  
 *# setting widgets* strip\_name = self.edit\_panel.strip\_name.GetLabel()  
 self.edit\_panel.volume.SetLabel(**f'{**self.edit\_panel.strips[strip\_name][2]**}%'**)  
 *# showing editing panel* self.edit\_panel.edit\_edit\_panel.Show()  
 self.edit\_panel.edit\_edit\_panel.Layout()  
 self.edit\_panel.edit\_panel.Layout()  
 self.edit\_panel.Layout()  
 self.Layout()  
  
 **def** editor\_edit(self, strip\_name, username, edit):  
 *"""  
 setting strip editors* **:param** *edit: boolean* **:param** *strip\_name: strip name* **:param** *username: username* **:return***: None  
 """* **if** edit:  
 self.edit\_panel.strip\_edit[strip\_name] = username  
 **else**:  
 **del** self.edit\_panel.strip\_edit[strip\_name]  
  
 **def** inc(self, event):  
 *"""  
 enc volume* **:param** *event: event or strip name* **:return***: None  
 """* strip\_name = self.edit\_panel.strip\_name.GetLabel()  
 volume = self.edit\_panel.strips[strip\_name][2] *# volume* **if** type(event) != str:  
 **if** self.edit\_panel.play\_audio.GetLabel() == **'play audio'**:  
 *# increasing volume* **if** volume < 100:  
 volume += 10  
 *# sending to server we want to inc volume* self.comm.send(ClientProtocol.inc\_volume(self.edit\_panel.strip\_name.GetLabel()))  
 *# telling logic to increase volume* wx.CallAfter(pub.sendMessage, **"inc"**, strip\_name=self.edit\_panel.strip\_name.GetLabel())  
 *# setting volume in panel* self.edit\_panel.volume.SetLabel(**f'{**volume**}%'**)  
 self.edit\_panel.edit\_edit\_panel.Layout()  
 self.edit\_panel.edit\_panel.Layout()  
 self.edit\_panel.Layout()  
 self.Layout()  
 **else**:  
 strip\_name = event  
 volume = self.edit\_panel.strips[strip\_name][2] *# volume* volume += 10  
 self.edit\_panel.strips[strip\_name][2] = volume  
  
 **def** dec(self, event):  
 *"""  
 dec volume* **:param** *event: event or strip name* **:return***: None  
 """* strip\_name = self.edit\_panel.strip\_name.GetLabel()  
 volume = self.edit\_panel.strips[strip\_name][2] *# volume* **if** type(event) != str:  
 **if** self.edit\_panel.play\_audio.GetLabel() == **'play audio'**:  
 *# increasing volume* **if** volume > 0:  
 volume -= 10  
 *# sending to server we want to dec volume* self.comm.send(ClientProtocol.dec\_volume(self.edit\_panel.strip\_name.GetLabel()))  
 *# telling logic to increase volume* wx.CallAfter(pub.sendMessage, **"dec"**, strip\_name=self.edit\_panel.strip\_name.GetLabel())  
 *# setting volume in panel* self.edit\_panel.volume.SetLabel(**f'{**volume**}%'**)  
 self.edit\_panel.edit\_edit\_panel.Layout()  
 self.edit\_panel.edit\_panel.Layout()  
 self.edit\_panel.Layout()  
 self.Layout()  
 **else**:  
 strip\_name = event  
 volume = self.edit\_panel.strips[strip\_name][2] *# volume* volume -= 10  
 self.edit\_panel.strips[strip\_name][2] = volume  
  
 **def** check\_cut\_beginning(self, event):  
 *"""  
 checking cutting strip from beginning* **:param** *event: event* **:return***: None  
 """* strip\_name = self.edit\_panel.strip\_name.GetLabel()  
 cut\_time = self.edit\_panel.beginning.GetValue()  
 diff = self.edit\_panel.strips[strip\_name][3]\*1000 - cut\_time  
 **if** cut\_time > 0:  
 **if** diff > 99:  
 *# sending to server to cut strip* self.comm.send(ClientProtocol.shorten\_start(strip\_name, cut\_time))  
 *# telling logic to cut strip* wx.CallAfter(pub.sendMessage, **"beginning"**, strip\_name=strip\_name, time=cut\_time)  
 *# clearing intctrl* self.edit\_panel.beginning.SetValue(0)  
 *# cutting button* self.cut\_beginning(strip\_name, cut\_time, **True**)  
 **else**:  
 self.pop\_error(**'you cannot cut this amount from the strip'**)  
  
 **def** cut\_beginning(self, strip\_name, cut\_time, flag):  
 *"""  
 cutting string from the beginning* **:param** *flag: flag for button color* **:param** *strip\_name: strip name* **:param** *cut\_time: cut time* **:return***: None  
 """  
 # cutting* starting\_point = self.edit\_panel.strips[strip\_name][1] *# starting point* length = self.edit\_panel.strips[strip\_name][3] *# strip length  
 # creating new strip button* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* strip\_btn = wx.Button(self.edit\_panel.strips\_scroller, label=strip\_name,  
 size=(int((length - cut\_time / 1000) \* 50), 80), style=wx.NO\_BORDER)  
 wave = wx.Bitmap(**'wave.png'**) *# create wx.Bitmap object* strip\_btn.SetBitmap(wave)  
 **if** flag:  
 strip\_btn.SetBackgroundColour(**'#3D8B48'**)  
 **else**:  
 strip\_btn.SetBackgroundColour(**'#8B473D'**)  
 strip\_btn.SetFont(btn\_font)  
 cut\_time /= 1000  
 self.edit\_panel.strips[strip\_name][5].Clear(**True**)  
 self.edit\_panel.strips[strip\_name][5].AddSpacer(int((starting\_point + cut\_time) \* 50))  
 self.edit\_panel.strips[strip\_name][5].Add(strip\_btn, 0, wx.LEFT | wx.ALIGN\_LEFT)  
 self.edit\_panel.strips\_scroller.SetupScrolling()  
 *# adding new button* self.edit\_panel.strips[strip\_name][0] = strip\_btn  
 self.edit\_panel.strips[strip\_name][1] = starting\_point + cut\_time  
 self.edit\_panel.strips[strip\_name][3] = length - cut\_time  
 *# setting length widget* self.edit\_panel.strip\_length.SetLabel(**f'strip length: {**(length - cut\_time)\*1000//1/1000**} seconds'**)  
 self.Layout()  
 self.edit\_panel.Layout()  
 self.edit\_panel.edit\_panel.Layout()  
 *# binding button* strip\_btn.Bind(wx.EVT\_BUTTON, self.show\_edit\_panel)  
  
 **def** check\_cut\_end(self, event):  
 *"""  
 checking cutting strip from the end* **:param** *event: event* **:return***: None  
 """* strip\_name = self.edit\_panel.strip\_name.GetLabel()  
 cut\_time = self.edit\_panel.end.GetValue()  
 diff = self.edit\_panel.strips[strip\_name][3] \* 1000 - cut\_time  
 **if** cut\_time > 0:  
 **if** diff > 99:  
 *# sending to server to cut strip* self.comm.send(ClientProtocol.shorten\_end(strip\_name, cut\_time))  
 *# telling logic to cut strip* wx.CallAfter(pub.sendMessage, **"end"**, strip\_name=strip\_name, time=cut\_time)  
 *# clearing intctrl* self.edit\_panel.end.SetValue(0)  
 *# cutting button* self.cut\_end(strip\_name, cut\_time, **True**)  
 **else**:  
 self.pop\_error(**'you cannot cut this amount from the strip'**)  
  
 **def** cut\_end(self, strip\_name, cut\_time, flag):  
 *"""  
 cutting strip from end* **:param** *strip\_name: strip name* **:param** *cut\_time: cut time* **:param** *flag: flag for color* **:return***: None  
 """  
 # cutting* starting\_point = self.edit\_panel.strips[strip\_name][1] *# starting point* length = self.edit\_panel.strips[strip\_name][3] *# strip length  
 # creating new strip button* btn\_font = wx.Font(15, wx.FONTFAMILY\_ROMAN, wx.FONTSTYLE\_NORMAL, wx.FONTWEIGHT\_BOLD) *# font* strip\_btn = wx.Button(self.edit\_panel.strips\_scroller, label=strip\_name,  
 size=(int((length - cut\_time / 1000) \* 50), 80), style=wx.NO\_BORDER)  
 wave = wx.Bitmap(**'wave.png'**) *# create wx.Bitmap object* strip\_btn.SetBitmap(wave)  
 **if** flag:  
 strip\_btn.SetBackgroundColour(**'#3D8B48'**)  
 **else**:  
 strip\_btn.SetBackgroundColour(**'#8B473D'**)  
 strip\_btn.SetFont(btn\_font)  
 cut\_time /= 1000  
 self.edit\_panel.strips[strip\_name][5].Clear(**True**)  
 self.edit\_panel.strips[strip\_name][5].AddSpacer(int(starting\_point \* 50))  
 self.edit\_panel.strips[strip\_name][5].Add(strip\_btn, 0, wx.LEFT | wx.ALIGN\_LEFT)  
 self.edit\_panel.strips\_scroller.SetupScrolling()  
 *# adding new button* self.edit\_panel.strips[strip\_name][0] = strip\_btn  
 self.edit\_panel.strips[strip\_name][3] = length - cut\_time  
 *# setting length widget* self.edit\_panel.strip\_length.SetLabel(**f'strip length: {**(length - cut\_time)\*1000//1/1000**} seconds'**)  
 self.Layout()  
 self.edit\_panel.Layout()  
 self.edit\_panel.edit\_panel.Layout()  
  
 *# binding button* strip\_btn.Bind(wx.EVT\_BUTTON, self.show\_edit\_panel)  
  
 **def** show\_panel\_one(self, event):  
 *""" showing panel 1"""* self.log\_in\_panel.Show()  
 self.sign\_in\_panel.Hide()  
 self.Layout()  
 event.Skip()  
  
 **def** show\_panel\_two(self, event):  
 *""" showing panel 2"""* self.sign\_in\_panel.Show()  
 self.log\_in\_panel.Hide()  
 self.Layout()  
 event.Skip()  
  
 **def** show\_panel\_three(self, event):  
 *""" showing panel 3"""* self.home\_panel.Show()  
 self.sign\_in\_panel.Hide()  
 self.log\_in\_panel.Hide()  
 self.add\_file\_panel.Hide()  
 self.Layout()  
 **if** event:  
 event.Skip()  
  
 **def** show\_panel\_four(self, event):  
 *""" showing panel 4"""* self.add\_file\_panel.Show()  
 self.home\_panel.Hide()  
 self.Layout()  
 event.Skip()  
  
 **def** show\_panel\_five(self, event):  
 *""" showing panel 5"""* self.edit\_panel.Show()  
 self.add\_file\_panel.Hide()  
 self.home\_panel.Hide()  
 self.loading\_panel.Hide()  
 self.add\_strip\_panel.Hide()  
 self.Layout()  
 **if** event:  
 event.Skip()  
  
 **def** show\_panel\_six(self, event):  
 *""" showing panel 6"""* **if** self.edit\_panel.play\_audio.GetLabel() == **'play audio'**:  
 *# clearing file anf strip name* self.add\_strip\_panel.strip = **''** self.add\_strip\_panel.strip\_file = **''** *# showing and hiding panels* self.edit\_panel.Hide()  
 self.add\_strip\_panel.Show()  
 self.Layout()  
 **if** event:  
 event.Skip()  
  
 **def** show\_panel\_loading(self, event):  
 *""" showing loading panel"""* self.loading\_panel.Show()  
 self.edit\_panel.Hide()  
 self.add\_strip\_panel.Hide()  
 self.home\_panel.Hide()  
 self.Layout()  
 **if** event:  
 event.Skip()  
  
 **def** pop\_error(self, msg):  
 *"""  
 pops error message* **:param** *msg: error message* **:return***: None  
 """* wx.MessageBox(msg, **'Error'**, wx.OK)

## music\_strip.py

**from** pydub **import** AudioSegment  
**import** os  
  
  
**class** MusicStrip:  
 *"""  
 class to make edits in music and create a strip object  
 """* **def** \_\_init\_\_(self, file, name, starting\_point, volume, path):  
 *"""  
 creates a strip object* **:param** *file: file in bytes* **:param** *name: strip name* **:param** *path: path to save file* **:param** *starting\_point: starting time* **:param** *volume: volume  
 """* self.name = name *# strip name* self.path = path *# strip path* **if** file: *# client  
 # saving file* **with** open(self.path+**'\\'**+name+**'.wav'**, **'wb'**) **as** strip:  
 strip.write(file)  
 self.file = AudioSegment.from\_wav(self.path+**'\\'**+name+**'.wav'**) *# pydub file* self.length = self.file.duration\_seconds *# length of file in seconds* self.muted = **False** *# if strip is muted* self.volume = volume *# strip volume* self.starting\_point = starting\_point *# starting point of strip* **def** trim\_start(self, time):  
 *"""  
 trimming from start* **:param** *time: time to trim* **:return***: None  
 """* self.file = self.file[time:] *# trimming file* self.length = self.file.duration\_seconds *# updating time* self.starting\_point += time/1000 *# updating starting point* **def** trim\_end(self, time):  
 *"""  
 trimming from end* **:param** *time: time to trim* **:return***: None  
 """* self.file = self.file[:-time] *# trimming file* self.length = self.file.duration\_seconds *# updating time* **def** set\_strip(self, file, starting\_point, volume):  
 *"""  
 setting strip with parameters* **:param** *file: music file in bytes* **:param** *starting\_point: starting point* **:param** *volume: volume* **:return***: None  
 """  
 # saving file* **with** open(self.path+**'\\'** + self.name + **'.wav'**, **'wb'**) **as** strip:  
 strip.write(file)  
  
 self.file = AudioSegment.from\_wav(**'client\_music\\'** + self.name + **'.wav'**) *# pydub file* self.length = self.file.duration\_seconds *# length of file in seconds* self.volume = volume *# strip volume* self.starting\_point = starting\_point *# starting point of strip* **def** save\_file(self):  
 *"""  
 saving strip with volume* **:return***: None  
 """* self.file.export(self.path+**'\\'** + self.name + **'.wav'**, format=**'wav'**) *# export strip audio file***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
  
 **pass**

## server\_com.py

**import** socket  
**import** threading  
**import** select  
**import** random  
**from** server\_protocol **import** ServerProtocol  
  
  
**class** ServerCom:  
 *"""  
 server communication  
 """* **def** \_\_init\_\_(self, port, msg\_q, encrypt):  
 *"""  
 initializing object parameters and calling the thread main loop* **:param** *port: communication port* **:param** *msg\_q: queue for messages* **:param** *encrypt: encrypt decrypt object  
 """* self.running = **True** self.my\_socket = **None** self.port = port  
 self.msg\_q = msg\_q  
 self.encrypt = encrypt  
 self.open\_clients = {} *# dictionary of open clients  
  
 # calling main\_loop thread* threading.Thread(target=self.\_main\_loop).start()  
  
 **def** \_main\_loop(self):  
 *"""  
 creating server socket starts to accept new clients  
 putting their messages in msg\_q and calling \_diffie() for new clients* **:return***: None  
 """* self.my\_socket = socket.socket()  
 self.my\_socket.bind((**'0.0.0.0'**, self.port))  
 self.my\_socket.listen(5)  
  
 **while** self.running:  
 rlist, wlist, xlist = select.select([self.my\_socket] + list(self.open\_clients.keys()), [], [], 0.1)  
 **for** current\_socket **in** rlist:  
 **if** current\_socket **is** self.my\_socket:  
 client, addr = self.my\_socket.accept()  
 print(**f"{**addr[0]**} - connected"**)  
 **if** addr[0] **in** self.open\_clients.values(): *# checking if ip is already connected to prevent dos* self.\_disconnect\_client(client)  
 **elif** addr[0] **not in** self.encrypt.encrypt\_keys.keys(): *# checking if to exchange keys  
 # calling \_diffie() thread to exchange key with client* threading.Thread(target=self.\_diffie, args=(client, addr,)).start()  
 **else**:  
 self.open\_clients[client] = addr[0] *# adding client to open clients* **elif** current\_socket **in** self.open\_clients.keys():  
 **try**:  
 data\_len = current\_socket.recv(5).decode() *# getting data length* data = current\_socket.recv(int(data\_len)) *# getting data  
 # decrypting data* data = self.encrypt.decrypt(data, self.open\_clients[current\_socket]).decode()  
 **if** data[:2] == **'16'**: *# if receive file* proto = int(data[:2]) *# message protocol* file\_name = data[2:] *# file name  
  
 # receiving starting point* data\_len = current\_socket.recv(5).decode()  
 start\_time = current\_socket.recv(int(data\_len))  
 *# decrypting data* start\_time = self.encrypt.decrypt(start\_time, self.open\_clients[current\_socket]).decode()  
 start\_time = start\_time[2:] *# removing protocol  
  
 # receiving volume* data\_len = current\_socket.recv(5).decode()  
 volume = current\_socket.recv(int(data\_len))  
 *# decrypting data* volume = self.encrypt.decrypt(volume, self.open\_clients[current\_socket]).decode()  
 volume = volume[2:] *# removing protocol  
  
 # receiving file* data = self.recv\_file(file\_name, start\_time, volume, current\_socket, proto)  
  
 **except** Exception **as** e:  
 print(**"ServerCom - \_main\_loop. server\_com.py:84 "** + str(e))  
 self.\_disconnect\_client(current\_socket)  
 **else**:  
 **if** data == **""**:  
 self.\_disconnect\_client(current\_socket)  
 **else**:  
 **if** current\_socket **in** self.open\_clients.keys(): *# if socket open clients* **if** type(data) == str: *# if not receive file* data = ServerProtocol.unpack(data)  
 self.msg\_q.put((self.open\_clients[current\_socket], data))  
  
 **def** \_diffie(self, client, addr):  
 *"""  
 exchanging key with client using Diffie Hellman protocol* **:param** *client: client socket* **:param** *addr: ip address of client* **:return***: the encryption key  
 """* p = 5195977  
 g = 35125  
  
 private\_a = random.randint(1, 9999) *# generating random private number* A = (g \*\* private\_a) % p *# number to exchange with client  
  
 # sending A and receiving B* **try**:  
 *# sending A* message = str(A)  
 client.send((str(len(message))+message).encode())  
  
 *# receiving B* length = client.recv(1).decode() *# getting message length* B = int(client.recv(int(length)).decode())  
  
 **except** Exception **as** e:  
 print(**'ServerCom \_diffie - '**, e)  
  
 **else**:  
 *# calculating key* key = (B \*\* private\_a) % p  
  
 print(**f'{**addr[0]**}, key-{**key**}'**)  
 self.open\_clients[client] = addr[0] *# adding client to open clients* self.encrypt.add\_key(str(key), addr[0]) *# adding key to encryption object* **def** send(self, ips, msg):  
 *"""  
 sends msg to list of sockets* **:param** *ips: list of ips* **:param** *msg: message to send* **:return***: None  
 """  
  
 # sending message to all clients* **for** ip **in** ips:  
 soc = self.find\_socket\_by\_ip(ip)  
  
 **try**:  
 enc = self.encrypt.encrypt(msg, self.open\_clients[soc])  
 *# sending length of message* soc.send(str(len(enc)).zfill(5).encode())  
 *# sending message* soc.send(enc)  
 **except** Exception **as** e:  
 print(**'serverCom send - '**, e)  
 self.\_disconnect\_client(soc)  
  
 **def** \_disconnect\_client(self, socket):  
 *"""  
 disconnects socket and removes socket from open\_clients* **:param** *socket: socket to disconnect* **:return***: None  
 """  
 # telling server to delete username* self.msg\_q.put((self.open\_clients[socket], [0]))  
 *# removing socket* **del** self.open\_clients[socket]  
 socket.close() *# closing socket* **def** find\_socket\_by\_ip(self, ip):  
 *"""  
 returning socket from ip* **:param** *ip: ip* **:return***: socket  
 """* socket = **None  
 for** soc **in** self.open\_clients.keys():  
 **if** self.open\_clients[soc] == ip:  
 socket = soc  
 **break  
 return** socket  
  
 **def** send\_file(self, ip, file\_path):  
 *"""  
 sending file to client* **:param** *ip: ip to send to* **:param** *file\_path: file path* **:return***: None  
 """* socket = self.find\_socket\_by\_ip(ip) *# socket of client  
 # getting file* **with** open(file\_path, **'rb'**) **as** strip:  
 file = strip.read()  
 enc = self.encrypt.encrypt(file, self.open\_clients[socket])  
  
 length = len(enc)  
 print(**f'length - {**length**}'**)  
 **try**:  
 socket.send(str(len(enc)).zfill(8).encode()) *# sending file length* **except** Exception **as** e:  
 self.\_disconnect\_client(socket)  
 print(**'ServerCom send\_file - '**, e)  
 **else**:  
 socket.sendall(enc)  
  
 **def** recv\_file(self, file\_name, start\_time, volume, socket, proto):  
 *"""  
 receiving music file from client* **:param** *file\_name: file name* **:param** *proto: protocol* **:param** *socket: client socket* **:param** *start\_time: start time* **:param** *volume: volume* **:return***: None  
 """* data = **""** *# data to return* **try**:  
 *# getting file bytes length* length = int(socket.recv(8).decode())  
 print(**f'length - {**length**}'**)  
 *# receiving file* file = bytearray()  
 **while** len(file) < length:  
 slice = length - len(file)  
 **if** slice > 2048:  
 file.extend(socket.recv(2048))  
 **else**:  
 file.extend(socket.recv(slice))  
 **break** print(**f'length2 - {**len(file)**}'**)  
 **except** Exception **as** e:  
 self.\_disconnect\_client(socket)  
 print(**'serverCom recv\_file - '**, e)  
 **else**:  
 file = self.encrypt.decrypt(file.decode(), self.open\_clients[socket])  
 data = [proto, file\_name, start\_time, volume, file]  
 *# returning data* **return** data

## server\_logic.py

**from** server\_com **import** ServerCom  
**import** queue  
**from** encrypt **import** Encrypt  
**from** data\_base **import** DB  
**import** threading  
**from** server\_protocol **import** ServerProtocol  
**from** edit\_group **import** EditGroup  
**import** os  
**import** time  
  
  
**class** ServerLogic:  
 *"""server logic"""* **def** \_\_init\_\_(self):  
 *# encrypt object* self.cp = Encrypt()  
 *# queue for messages* self.msg\_q = queue.Queue()  
 *# communication object* self.comm = ServerCom(2000, self.msg\_q, self.cp)  
 *# editing groups = {file name : EditGroup}* self.editing\_groups = {}  
 *# usernames = {ip, username}* self.usernames = {}  
  
 *# handle msgs* threading.Thread(target=self.handle\_msgs).start()  
  
 **def** handle\_msgs(self):  
 *"""  
 handling msgs from server* **:return***: None  
 """* **while True**:  
 ip, msg = self.msg\_q.get()  
  
 **if** msg[0] == -1: *# close edit group* **del** self.editing\_groups[msg[1]]  
 **if** msg[0] == 0: *# user disconnected* **del** self.cp.encrypt\_keys[ip]  
 **if** ip **in** self.usernames.keys():  
 **del** self.usernames[ip]  
 **elif** msg[0] == 1: *# log in* self.log\_in(ip, msg[1], msg[2])  
 **elif** msg[0] == 2:  
 self.sign\_in(ip, msg[1], msg[2])  
 **elif** msg[0] == 3: *# create edit file request* self.create\_edit\_file(ip, msg[1])  
 **elif** msg[0] == 4: *# edit file* self.edit\_file(ip, msg[1])  
 **elif** msg[0] == 5:  
 **pass  
 elif** msg[0] == 6:  
 **pass  
 elif** msg[0] == 7:  
 **pass  
 elif** msg[0] == 8:  
 **pass  
 elif** msg[0] == 9:  
 **pass  
 elif** msg[0] == 10:  
 **pass  
 elif** msg[0] == 11:  
 **pass  
 elif** msg[0] == 12:  
 **pass  
 elif** msg[0] == 13:  
 **pass  
 elif** msg[0] == 14:  
 **pass  
 elif** msg[0] == 15:  
 **pass  
 elif** msg[0] == 16:  
 **pass  
 elif** msg[0] == 17:  
 **pass  
  
 def** log\_in(self, ip, password, username):  
 *"""  
 checks if password and username are correct* **:param** *ip: client ip* **:param** *password: client password* **:param** *username: client username* **:return***: None  
 """* db = DB() *# data base object  
  
 # checking if user is already logged in* **if** username **not in** self.usernames.values():  
 password = str(Encrypt.hash(password))  
  
 *# checking if password is correct* **if** db.check\_password(username, password):  
 *# sending correct* self.comm.send([ip], **'05'**)  
 *# adding username to logged users* self.usernames[ip] = username  
  
 *# sending the names of users to client* usernames = [username] + db.get\_users(username)  
 **if** len(usernames) > 0:  
 self.comm.send([ip], ServerProtocol.usernames(usernames))  
  
 *# sending edit files* edit\_files = db.get\_user\_files(username)  
 **if** len(edit\_files) > 0:  
 self.comm.send([ip], ServerProtocol.edit\_files(edit\_files))  
  
 **else**:  
 *# sending password or username are incorrect* self.comm.send([ip], **'01'**)  
 **else**:  
 *# sending to client that the user is already logged in* self.comm.send([ip], **'02'**)  
 *# closing data base* db.close\_db()  
  
 **def** sign\_in(self, ip, password, username):  
 *"""  
 adding client if not exists* **:param** *ip: ip* **:param** *password: client password* **:param** *username: client username* **:return***: None  
 """* db = DB() *# data base object* password = str(Encrypt.hash(password))  
 added = db.add\_user(username, password)  
 **if** added:  
 *# adding username to logged users* self.usernames[ip] = username  
  
 *# sending user was added* self.comm.send([ip], **'04'**)  
  
 *# sending the names of users to client* usernames = [username] + db.get\_users(username)  
 **if** len(usernames) > 0:  
 self.comm.send([ip], ServerProtocol.usernames(usernames))  
  
 *# sending edit files* edit\_files = db.get\_user\_files(username)  
 **if** len(edit\_files) > 0:  
 self.comm.send([ip], ServerProtocol.edit\_files(edit\_files))  
  
 *# sending to all connected clients the new added user* self.send\_add\_user(username, ip)  
  
 **else**:  
 *# sending user already exists* self.comm.send([ip], **'03'**)  
 *# closing data base* db.close\_db()  
  
 **def** edit\_file(self, ip, file\_name):  
 *"""  
 sends to client details about edit file* **:param** *ip: client ip* **:param** *file\_name: file name* **:return***: None  
 """  
  
 # checking if there's already an EditGroup object* **if** file\_name **in** self.editing\_groups.keys():  
 *# adding client to edit group* self.editing\_groups[file\_name].add\_editor(self.comm.find\_socket\_by\_ip(ip), self.usernames[ip], ip)  
 **else**:  
 *# creating an edit group object* self.editing\_groups[file\_name] = EditGroup(self.comm.find\_socket\_by\_ip(ip), self.usernames[ip]  
 , file\_name, self.msg\_q, self.cp, ip)  
 *# removing client from open clients* **del** self.comm.open\_clients[self.comm.find\_socket\_by\_ip(ip)]  
  
 **def** create\_edit\_file(self, ip, file\_name):  
 *"""  
 sending a message to client if file already exists and adding file if it does'nt exist* **:param** *ip: client ip* **:param** *file\_name: file name* **:return***: None  
 """* db = DB() *# data base object* file = file\_name.split(**';'**)[0]  
 group\_members = file\_name.split(**';'**)[1]  
  
 *# trying to add file to data base* **if** db.add\_file(file, group\_members):  
 *# creating folder for edit file* path = os.path.join(**'groups\_files'**, file\_name)  
 os.mkdir(path)  
 *# creating folder for edit file undo* path = os.path.join(**'groups\_files'**, **f'{**file\_name**}\\undo'**)  
 os.mkdir(path)  
 *# sending added file* self.send\_add\_file(file\_name, ip)  
 self.edit\_file(ip, file\_name)  
 **else**:  
 *# sending to client that file already exists* self.comm.send([ip], **'32'**)  
  
 **def** send\_add\_user(self, username, ip):  
 *"""  
 sending to all connected clients the new added user* **:param** *username: username* **:param** *ip: client ip* **:return***: None  
 """* **if** len(self.usernames) > 1:  
 *# sending to all connected users to add new user* ips = list(self.usernames.keys())  
 ips.remove(ip) *# removing client ip* send\_ips = []  
 **for** i **in** ips: *# finding which ips to send to* **if** i **in** self.comm.open\_clients.values():  
 send\_ips.append(i)  
 self.comm.send(ips, ServerProtocol.added\_user(username))  
  
 **def** send\_add\_file(self, file\_name, ip):  
 *"""  
 sending to all connected users the new added file* **:param** *file\_name: file name* **:param** *ip: client ip* **:return***: None  
 """* **if** len(self.usernames) > 1:  
 ips = list(self.usernames.keys())  
 ips.remove(ip) *# removing client ip* group\_members = file\_name.split(**';'**)[1].split(**","**)  
 send\_ips = []  
 **for** i **in** ips: *# finding which ips to send to* **if** self.usernames[i] **in** group\_members **and** i **in** self.comm.open\_clients.values():  
 send\_ips.append(i)  
 **if** len(send\_ips) > 0:  
 *# sending to clients file to add* self.comm.send(send\_ips, ServerProtocol.added\_file(file\_name))  
  
  
**def** main():  
 *""" main server"""* logic = ServerLogic() *# logic object***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

## server\_protocol.py

**from** server\_com **import** ServerCom  
**import** queue  
**from** encrypt **import** Encrypt  
**from** data\_base **import** DB  
**import** threading  
**from** server\_protocol **import** ServerProtocol  
**from** edit\_group **import** EditGroup  
**import** os  
**import** time  
  
  
**class** ServerLogic:  
 *"""server logic"""* **def** \_\_init\_\_(self):  
 *# encrypt object* self.cp = Encrypt()  
 *# queue for messages* self.msg\_q = queue.Queue()  
 *# communication object* self.comm = ServerCom(2000, self.msg\_q, self.cp)  
 *# editing groups = {file name : EditGroup}* self.editing\_groups = {}  
 *# usernames = {ip, username}* self.usernames = {}  
  
 *# handle msgs* threading.Thread(target=self.handle\_msgs).start()  
  
 **def** handle\_msgs(self):  
 *"""  
 handling msgs from server* **:return***: None  
 """* **while True**:  
 ip, msg = self.msg\_q.get()  
  
 **if** msg[0] == -1: *# close edit group* **del** self.editing\_groups[msg[1]]  
 **if** msg[0] == 0: *# user disconnected* **del** self.cp.encrypt\_keys[ip]  
 **if** ip **in** self.usernames.keys():  
 **del** self.usernames[ip]  
 **elif** msg[0] == 1: *# log in* self.log\_in(ip, msg[1], msg[2])  
 **elif** msg[0] == 2:  
 self.sign\_in(ip, msg[1], msg[2])  
 **elif** msg[0] == 3: *# create edit file request* self.create\_edit\_file(ip, msg[1])  
 **elif** msg[0] == 4: *# edit file* self.edit\_file(ip, msg[1])  
 **elif** msg[0] == 5:  
 **pass  
 elif** msg[0] == 6:  
 **pass  
 elif** msg[0] == 7:  
 **pass  
 elif** msg[0] == 8:  
 **pass  
 elif** msg[0] == 9:  
 **pass  
 elif** msg[0] == 10:  
 **pass  
 elif** msg[0] == 11:  
 **pass  
 elif** msg[0] == 12:  
 **pass  
 elif** msg[0] == 13:  
 **pass  
 elif** msg[0] == 14:  
 **pass  
 elif** msg[0] == 15:  
 **pass  
 elif** msg[0] == 16:  
 **pass  
 elif** msg[0] == 17:  
 **pass  
  
 def** log\_in(self, ip, password, username):  
 *"""  
 checks if password and username are correct* **:param** *ip: client ip* **:param** *password: client password* **:param** *username: client username* **:return***: None  
 """* db = DB() *# data base object  
  
 # checking if user is already logged in* **if** username **not in** self.usernames.values():  
 password = str(Encrypt.hash(password))  
  
 *# checking if password is correct* **if** db.check\_password(username, password):  
 *# sending correct* self.comm.send([ip], **'05'**)  
 *# adding username to logged users* self.usernames[ip] = username  
  
 *# sending the names of users to client* usernames = [username] + db.get\_users(username)  
 **if** len(usernames) > 0:  
 self.comm.send([ip], ServerProtocol.usernames(usernames))  
  
 *# sending edit files* edit\_files = db.get\_user\_files(username)  
 **if** len(edit\_files) > 0:  
 self.comm.send([ip], ServerProtocol.edit\_files(edit\_files))  
  
 **else**:  
 *# sending password or username are incorrect* self.comm.send([ip], **'01'**)  
 **else**:  
 *# sending to client that the user is already logged in* self.comm.send([ip], **'02'**)  
 *# closing data base* db.close\_db()  
  
 **def** sign\_in(self, ip, password, username):  
 *"""  
 adding client if not exists* **:param** *ip: ip* **:param** *password: client password* **:param** *username: client username* **:return***: None  
 """* db = DB() *# data base object* password = str(Encrypt.hash(password))  
 added = db.add\_user(username, password)  
 **if** added:  
 *# adding username to logged users* self.usernames[ip] = username  
  
 *# sending user was added* self.comm.send([ip], **'04'**)  
  
 *# sending the names of users to client* usernames = [username] + db.get\_users(username)  
 **if** len(usernames) > 0:  
 self.comm.send([ip], ServerProtocol.usernames(usernames))  
  
 *# sending edit files* edit\_files = db.get\_user\_files(username)  
 **if** len(edit\_files) > 0:  
 self.comm.send([ip], ServerProtocol.edit\_files(edit\_files))  
  
 *# sending to all connected clients the new added user* self.send\_add\_user(username, ip)  
  
 **else**:  
 *# sending user already exists* self.comm.send([ip], **'03'**)  
 *# closing data base* db.close\_db()  
  
 **def** edit\_file(self, ip, file\_name):  
 *"""  
 sends to client details about edit file* **:param** *ip: client ip* **:param** *file\_name: file name* **:return***: None  
 """  
  
 # checking if there's already an EditGroup object* **if** file\_name **in** self.editing\_groups.keys():  
 *# adding client to edit group* self.editing\_groups[file\_name].add\_editor(self.comm.find\_socket\_by\_ip(ip), self.usernames[ip], ip)  
 **else**:  
 *# creating an edit group object* self.editing\_groups[file\_name] = EditGroup(self.comm.find\_socket\_by\_ip(ip), self.usernames[ip]  
 , file\_name, self.msg\_q, self.cp, ip)  
 *# removing client from open clients* **del** self.comm.open\_clients[self.comm.find\_socket\_by\_ip(ip)]  
  
 **def** create\_edit\_file(self, ip, file\_name):  
 *"""  
 sending a message to client if file already exists and adding file if it does'nt exist* **:param** *ip: client ip* **:param** *file\_name: file name* **:return***: None  
 """* db = DB() *# data base object* file = file\_name.split(**';'**)[0]  
 group\_members = file\_name.split(**';'**)[1]  
  
 *# trying to add file to data base* **if** db.add\_file(file, group\_members):  
 *# creating folder for edit file* path = os.path.join(**'groups\_files'**, file\_name)  
 os.mkdir(path)  
 *# creating folder for edit file undo* path = os.path.join(**'groups\_files'**, **f'{**file\_name**}\\undo'**)  
 os.mkdir(path)  
 *# sending added file* self.send\_add\_file(file\_name, ip)  
 self.edit\_file(ip, file\_name)  
 **else**:  
 *# sending to client that file already exists* self.comm.send([ip], **'32'**)  
  
 **def** send\_add\_user(self, username, ip):  
 *"""  
 sending to all connected clients the new added user* **:param** *username: username* **:param** *ip: client ip* **:return***: None  
 """* **if** len(self.usernames) > 1:  
 *# sending to all connected users to add new user* ips = list(self.usernames.keys())  
 ips.remove(ip) *# removing client ip* send\_ips = []  
 **for** i **in** ips: *# finding which ips to send to* **if** i **in** self.comm.open\_clients.values():  
 send\_ips.append(i)  
 self.comm.send(ips, ServerProtocol.added\_user(username))  
  
 **def** send\_add\_file(self, file\_name, ip):  
 *"""  
 sending to all connected users the new added file* **:param** *file\_name: file name* **:param** *ip: client ip* **:return***: None  
 """* **if** len(self.usernames) > 1:  
 ips = list(self.usernames.keys())  
 ips.remove(ip) *# removing client ip* group\_members = file\_name.split(**';'**)[1].split(**","**)  
 send\_ips = []  
 **for** i **in** ips: *# finding which ips to send to* **if** self.usernames[i] **in** group\_members **and** i **in** self.comm.open\_clients.values():  
 send\_ips.append(i)  
 **if** len(send\_ips) > 0:  
 *# sending to clients file to add* self.comm.send(send\_ips, ServerProtocol.added\_file(file\_name))  
  
  
**def** main():  
 *""" main server"""* logic = ServerLogic() *# logic object***if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()