



ספר פרויקט גמר 2024

בית ספר: תיכון שש שנתי הנדסאים הרצליה

שם העבודה: Connectify

שם התלמיד: קוין פטריק טולי

ת.ז: 326968864

שם המנחה: אופיר שביט

תאריך הגשה: 18/5/24

תוכן עניינים

3	מבוא
3	הגדרת המוצר
3	סקירת פתרונות קיימים
6	תכנון וניהול לו"ז לפיתוח המערכת:
8	תיחום הפרויקט:
8	סקירת חולשות ואיומים:
10	פירוט יכולות:
16	ארכיטקטורה
16	ממשק גרפי
23	סביבת עבודה
24	מוסד הנתונים:
29	קשר בין הטבלאות:
29	הגדרת פרוטוקול התקשרות
31	מימוש הפרויקט
31	סקירת כל המחלקות המרכיבים את הפרויקט:
103	מסמך בדיקות מלא:
106	מדריך למשתמש
107	אופן ההפעלה:
114	רפלקציה
117	ביבליוגרפיה
118	נספחים

מבוא

הגדרת המוצר

אפליקציה בה תהיה אפשרות לתקשר עם החברים שלך או להכיר אנשים בסביבת הרשת היא משהו שהרבה אנשים עושים ואוהבים לעשות. אפשרות לתקשר איתם בעזרת הודעות טקסט, שיחות קוליות, בניית קבוצות של כמה וכמה חברים שכולם יכולים לשוחח ביחד. סביבה כזו כאשר היא מואבטחת ומאוד נגישה הוא פתרון שהרבה אנשים מחפשים.

הפרויקט יהיה אפליקציה הנגישה ללקוח שהוא יוכל להתחבר אליו בעזרת פרטים אם הוא כבר נירשם. אם הוא עדיין לא נירשם הוא יוכל להירשם דרך דף הבית. תהיה אפשרות ליצור קבוצות וצ'אטים עם כמה אנשים, יכולת לתקשר בעזרת צ'אט ושיחות קוליות. יהיה לך אפשרות להוסיף חברים ולחסום אנשים שאינך רוצה לשוחח איתם ולנהל את המשתמש שלך כיצד שתראה.

הפרויקט יהיה בנוי כאפליקציה בסביבת העובדה python, תוכל להתחבר למשתמש שלך או להירשם ולפעול לפי רצונך.

מטרת הפרויקט הוא להיות רשת חברתית נגישה ובטוחה בה כולם יוכלו לתקשר עם מי שירצו מתי שירצו על ידי אמצעים קוליים ואמצעי טקסט וגם מימוש וידאו. בנוסף תהיה באפליקציה אזור של פלייליסט בה הלקוח יוכל לשמוע שירים ולחפש שירים לפי הטעם שלו.

סקירת פתרונות קיימים

האפליקציה הוא למעשה רשת חברתית ומחולקת לכמה דפים עיקריים או נושאים שהאפליקציה עוסק בו. האפליקציה מתחלק לכניסה לאפליקציה, או להירשמות אליו. לאחר הכניסה אפשר להגיע לדף הראשי בו תוכל לבחור מכמה אפשרויות מה תרצה לעשות. זה החלק הראשי מתחלק ל-2 חלקים החלק הראשון הוא ההגדרות של המשתמש והחלק הנוסף הוא החלק של התקשורת בין שני משתמשים בין אם זה שיחות קוליות או צ'אט ביניהם.

נושא ראשון – כניסה אל האפליקציה או הירשמות אל האפליקציה:

כניסה והירשמות קיים למעשה כמעט בכל אפליקציה אינטרנט קיום. אפשר להירשם לאפליקציה בין אם זה עם כתובת אימייל, שם משתמש, סיסמא. ואפשרויות רבות בין אם זה גיל וכו...

פייסבוק הוא רשת חברתית מפורסמת בעולם המאפשרת למשתמשים ליצור פרופיל אישי, להתחבר עם חברים, לשתף תוכן, להצטרף לקבוצות וליצור אירועים וכמובן להירשם ולהתחבר אל אפליקציה החברה.

Facebook הוא למעשה אתר רשת חברתית שגם מישהו שיש לו משתמש יכול להיכנס אל האתר דרך דף ה־log in או להירשם דרך דף ה־sign up משהו שישימש בדיוק באתר זה. בנוסף

דרך הכניסה של פייסבוק אפשר יש מגנון שכיחה של סיסמא. כאשר אדם נכנס עם המשתמש או האימייל ואינו זוכר את הסיסמא הוא יכול לקבל קוד באימייל בשביל לשנות סיסמא או לינק, דבר שיפעל רק בווריאציה שונה באפליקציה שלי.

נושא שני – התקשורות בין משתמשים:

אחד הדוגמאות הפופולריות לאפליקציה שמשתמש בתקשורת בין משתמשים הוא "WhatsApp".

WhatsApp היא אפליקציה להודעות טקסט ושיחות קוליות שפותחה על ידי בריאן אקטון וג'ון קומ בשנת 2009.

אופן התקשורת ב-WhatsApp כולל את הבא:

הודעות טקסט: משתמשים יכולים לשלוח הודעות טקסט בזמן אמת לאנשים ברשימת הקשר שלהם. ההודעות מתקבלות מיד וניתן להגיב להן.

שיחות קוליות: חוץ מהודעות טקסט, המשתמשים יכולים גם לבצע שיחות קוליות באמצעות האפליקציה. זה מאפשר שיחה בזמן אמת לחברים ולמשפחה.

שיתוף קבוצות: WhatsApp מאפשרת למשתמשים ליצור קבוצות ולשתף בהן הודעות טקסט ותמונות עם קבוצת אנשים. זה נהיה שימושי במיוחד לתכנון אירועים או לתקשורת קבוצתית.

הודעות מאובטחות: WhatsApp משתמשת בהצפנה קצתקצנה על מנת להבטיח את פרטיות המשתמשים, מה שאומץ בכדי לשמור על סודיות התקשורת בין המשתמשים.

מיקום גיאוגרפי: האפליקציה מאפשרת למשתמשים לשלוח את מיקומם הגיאוגרפי לאנשים ברשימת הקשר, מה שיכול להיות שימושי למציאת מיקום או תקשורת ברחוב.

WhatsApp מספקת פלטפורמה נגיעה לתקשורת בין משתמשים בזמן אמת ומאובטחת, והיא זמינה לשימוש על מגוון מכשירים, כולל מחשבים, סמארטפונים וטאבלטים.

חלק מאמצעי התקשורת בהם וואטסאפ משתמש ישומשו גם באפליקציה בשביל חווית המשתמש.

נושא שלישי – הגדרות המשתמש:

באפליקציה יהיה הגדרות מובנות בשביל הנגישות של המשתמש הגדרות עיצוביות והגדרות תכניות. זהו מאפיין שקיים גם הוא בכמעט כל תוכנה הוא אפליקציה על מנת לשפר את החוויה של המשתמש מכיוון שכל אדם מותאם למשהו אחר ונוח לו משהו אחר.

דוגמא טובה לתוכנה שמשתמשת בfeatures הזה היא תוכנת ה-ios.

ההגדרות (Setting) במערכת ההפעלה iOS של Apple הן אזור חשוב שמספק לסמארטפון או הטאבלט שלך את היכולת להתאים את ההתנהגות וההפרטים האישיים של המכשיר. באמצעות ההגדרות, אפשר לשנות את הפרמטרים השונים שמספיעים על השימוש במכשיר, להתאים אותו לצרכים אישיים ולשלט במגוון אפשרויות.

בתוך ההגדרות, ניתן לקבוע את הרמת הבהירות של המסך, לשנות את הגדרות הרשתות האלוטיות, לנהל את החשבון האישי שלך ב-Apple ID, לבחור את הצלילים והרמקולים של המכשיר, להתנהל בכרטיס ה-SIM ועוד.

כמו כן, ההגדרות הן מקום חשוב לצפות במידע אודות המכשיר, לשדרג אותו לגרסה העדכנית ביותר של מערכת ההפעלה, ולבצע גיבויים על מנת לשמור על המידע החשוב שבמכשירך.

כמובן שההגדרות באפליקציה יהיה קצת שונות מההגדרות של apple משתמשת מכיוון שמדובר בתוכנת הפעלה של טלפון לעומת הגדרות של אפליקציה.

נושא רביעי – הפלייליסט של המשתמש:

יהיה למשתמש אזור אישי של פלייליסט שלו. בו הוא יכול להוסיף או להוריד שירים מהפלייליסט על פי רצונו ולשמע אותו. מאפיין זה הוא מאוד פופולרי כי המון אנשים אוהבים לשמוע מוזיקה ולשמור אותה בשבילם לאחר כן. תוכנות מאוד פופולריות המוכרות עם האפשרות הזו הם apple music ו-spotify.

spotify היא פלטפורמה דיגיטלית לשידור, חלוקה והאזנה למוזיקה ותכנים אודיו נוספים באינטרנט. ספוטיפיי מציעה ממשק משתמש ידידותי ומרכזי שמאפשר למשתמשים לחפש, לנווט וליצור פלייליסטים מוזיקליים ממגוון רחב של ספריות מוזיקליות. כמו כן, ספוטיפיי מציעה גם יכולת לשמוע מוזיקה במצב אופליין ולהפעיל אפשרויות רדיו ופודקאסטים מגוונים. לכל משתמש יש חשבון אישי שמאפשר לו ליצור ולנהל פלייליסטים מוזיקליים אישיים, לשמוע מוזיקה לפי טעמו האישי, לקבל המלצות מוזיקליות ממערכת ההמלצות החכמה של ספוטיפיי ולחקור אמנים, אלבומים ושירים מכל העולם. הפלטפורמה מציעה מגוון רחב של תוכן מוזיקלי, כולל מיליוני שירים מכל סוגי הז'אנרים והסגנונות, ומספקת חוויית האזנה איכותית ונוחה למשתמשים מכל רחבי העולם. Spotify מאפשרת גם לאמנים ולמוזיקאים להעלות את היצירות שלהם לפלטפורמה ולחלוק את מוזיקתם עם קהל המאזינים. בנוסף, Spotify מציעה אפשרות לשימוש ב-API המאפשר למפתחים ליצור אפליקציות ופתרונות תוכנה נוספים המשתמשים בנתוני המוזיקה והפודקאסטים של Spotify לצורך יצירת חוויות מוזיקליות ייחודיות ומותאמות אישית למשתמשים.

אומנם היישום לא יהיה מדויק למוצר spotify אך יהיה מאוד דומה. תיהיה אפשרות לשמוע שירים אך הם יהיו מרוכזים כפלייליסט אחד, לא תיהיה אפשרות לעלות קבצים אלא רק לחפש שירים שכבר קיימים בעזרת מנגנון חיפוש שנתון לאפליקציה.

תכנון וניהול לו"ז לפיתוח המערכת:

על מנת לפתח את האפליקציה שלי הייתי צריך ללמוד ספריית gui בשם pyqt5 הנמצאת בפיתון. לכן השלב הראשון בתכנון הלו"ז היה ללמוד את הבסיס ולהבין איך ליצור דפים פשוטים.

שלב ראשוני - למידת gui:

שלב 1: הגדרה והכנה

- הגדרת מטרות הפרויקט ודרישות ה-GUI.
- בחירת ספריות PyQt5 רלוונטיות.
- התקנת PyQt5 וביצוע בדיקת התקנה בסיסית.
- היכרות עם מושגי יסוד ב-PyQt5, כגון ווידג'טים, פריסות, איתותים ואירועים.

שלב 2: עיצוב ממשק המשתמש הבסיסי

- יצירת חלון ראשי ותכנון פריסת העמודות הבסיסית.
- עיצוב רכיבים עיקריים, כגון:
 - שדה טקסט להזנת הודעות.
 - חלון תצוגה להצגת הודעות.
 - רשימת משתמשים פעילים.
 - כפתורים לפעולות בסיסיות (שליחת הודעות, הצטרפות / עזיבה משרתים וכו').
- חיבור רכיבים לאיתותים ואירועים בסיסיים.

שלב 3: פונקציונליות בסיסית

- מימוש פונקציונליות בסיסית של שליחת הודעות והצגתן בחלון התצוגה.
- טיפול באירועי לחיצה על כפתורים וביצוע פעולות מתאימות (הצטרפות / עזיבה משרתים וכו').

- ניהול רשימת משתמשים פעילים ותצוגתה.

- שילוב טיפול בשגיאות ובחריגות.

שלב 4: שיפורים וחידושים

- הוספת אפשרויות עיצוב נוספות להתאמת ממשק המשתמש.

- שילוב אייקונים וגרפיקה.

- מימוש תכונות נוספות, כגון:

- ניהול ערוצים ושרתים.

- שליחת קבצים.

- התראות על הודעות חדשות.

שלב שני - חשיבה על המחלקות שאצטרך ויצירה שלהם:

בשלב השני אחרי הבניית gui רציתי להכין את כל המחלקות שאני אצטרך לניהול הפרויקט.

ביניהם פונקציות שינהלו את כניסת הלקוח לשרת ויציאתו, צילצולים, שיחות, סטריימינג.

על מנת לדעת בדיוק אילו מחלקות אני אצטרך רשמתי על דף את היכולות המרכזיות של הפרויקט ולאחר מכן חשבתי אילו מחלקות יעזרו לי להתמודד בקלות עם היכולות. מנגד ליכולות שאני יוצר לפרויקט מדובר גם בחלונות של ה GUI שלי - על מנת ליצור ווידג'טים(חלון או חלק מחלון) בqt5py אתה מכין מחלקה המתארת את החלון הזה לכן יש המון מחלקות המתארות חלונות.

שלב שלישי - יצירת פרוטוקול התקשורת וחיבור בין השרת ללקוח תוך שימוש במחלקות:

לאחר שני השלבים הראשונים רוב הקוד כבר בנוי - יש לנו GUI שפועל ברובו ואת המחלקות הנצרכות. בשלב זה יש לבנות את פרוטוקול התקשורת על מנת לבנות תקשורת בין השרת ללקוח כדי להעביר הודעות ומידע. לאחר בניית הפרוטוקול של שליחה וקבלת הודעות אפשר

להתחיל בחיבור עצמו - ניסוח הודעות, בקשות והתגובה של השרת לכל הודעה שהתקבלה -
תוך חיבור למחלקות הנדרשות.

שלב רביעי - גימור הפרויקט:

שלב זה מתרכז בשפשופים האחרונים בפרויקט. בשלב זה הפרויקט כבר אמור לעבוד מאוד טוב וצריכה להיות אפשרות להשתמש בכל היכולת שקבעתי. עכשיו מה שנשאר לעשות זה לסדר את הקוד, לראות שהכל עובד בצורה יעילה ואם לא לפעול בהתאם ובעיקר לדקדק את העיצוב ותיקון באגים סופיים.

תיחום הפרויקט:

אחד התחומים העיקריים שבו הפרויקט שלי עוסק הוא תחום הרשתות שכן אני יוצר שרת מרובה משתמשים שמסוגל לנהל שיחות וסטריימים בין המון לקוחות. בנוסף לכך הפרויקט שלי עוסק המון בעיצוב ה-GUI שכן חלק גדול מהקוד שלי מוקדש לעיצוב הנראה לעין של המסכים, תוך שימוש בספריית PYQT5 שעוסקת בעיקר ב-GUI אך נותנת מענה להרבה עיסוקים אחרים כולל תחום האודיו והוידאו. אבטחת מידע היא חלק מאוד חשוב שפרויקט שלי כי כן אני מבצע הצפנת aes לכל קטע מידע, בנוסף מתבצע החלפת מפתחות בכניסת כל לקוח לשרת בשילוב RSA. משתלבים אל תוך הפרויקט עוד הרבה מנגנוני אבטחה מה שמבטיחים על שימוש בטוח באפליקציה.

סקירת חולשות ואיומים:

SQL Injection: התקפה שבה משתמש זדוני מזריק קוד SQL זדוני לבקשות SQL של האפליקציה.

התבטאות: גניבת נתונים, שינוי נתונים, מחיקת נתונים, ביצוע פעולות לא מורשות.

פתרון: שימוש בפרמטרים מוכנים (Prepared Statements) והימנעות מהרכבת שאילתות SQL באופן ידני.

Brute Force Attack: זו התקפה בה התוקף מנסה את כל הצירופים האפשריים של תווים עד שהוא מוצא את הסיסמא הנכונה.

התבטאות: גניבת נתונים, שינוי נתונים, מחיקת נתונים, ביצוע פעולות לא מורשות.

פתרון: באפליקציה שלי אחד הפתרונות המאוד נגישים הוא פתרון ה-2FA או "אישור שני גורמים". איך זה עובד? באפליקציה שלי בדף ההגדרות יש למשתמש אופציה להדליק את הגדרת ה-2FA. הגדרה זו גורמת שבכל כניסה לאפליקציה המשתמש יצטרך להקיש את הפרטים הרגילים שלו אך גם את הקוד שהוא קיבל לאימייל באותו הרגע. מסך קוד האישור גם הוא פתוח למתקפת brute force לכן הפתרון שיישמתי באפליקציה הוא שיש עד 3 פעמים להקיש את הקוד הנכון, ברגע שהוקשו 3 קודים לא נכונים השרת עושה איפוס לתהליך.

Denial of Service - DoS: התקפה: תוקף מנסה להציף את השרת בבקשות מרובות במטרה לגרום לו להיות לא זמין.

התבטאות: השבתת השרת, האטת תגובת השרת, חוסר זמינות של השירותים.

פתרון: הגבלת מספר הבקשות לכתובת IP מסוימת. אם לקוח אחד שולח יותר מדי בקשות ומציף את השרת, השרת מגביל את קצב הבקשות מאותו לקוח.

Man-In-The-Middle - MITM: התקפה: תוקף משיג גישה לנתוני התקשורת בין הלקוח לשרת ומסוגל לקרוא או לשנות את הנתונים.

התבטאות: גניבת מידע, שינוי נתונים, התחזות, האזנה לתקשורת.

פתרון: אין פתרון מוחלט למניעת התקפה זו, אך שימוש בהצפנות יכול להקשות.

Social Engineering Attacks: התקפה: תוקף משתמש בטכניקות פסיכולוגיות להטעיה של משתמשים ולגרום להם לחשוף מידע רגיש או לבצע פעולות מסוימות.

התבטאות: גניבת מידע אישי, התחזות, שינוי נתונים, ביצוע פעולות לא מורשות.

פתרון: אי אפשר לדבר עם זרים בתוך האפליקציה. רק אנשים שאתה חבר שלהם או קשורים לחברים שלך יכולים לדבר איתך. בנוסף יש אפשרות בהגדרות לבטל תקשורת מוחלטת עם אנשים שאינם ברשימת החברים שלך.

פירוט יכולות:

השרת בפרויקט שלי הוא שרת בפייתון האחראי על התקשורת עם המשתמשים. הוא אחראי להביא להם את המידע הרצוי, לאפשר לתקשר אחד עם השני ולשמור את הנתונים הרלוונטיים שלהם במסד הנתונים.

יכולות השרת:

- שליחת מידע ללקוח
- שמירת מידע בתוך מסד הנתונים
- קבלת מידע מהלקוח
- הוצאת מידע ממסד הנתונים
- טיפול במידע של מסד הנתונים
- ניהול שיחות
- ניהול צלולים
- ניהול סטרימים

יכולות הלקוח:

- פועל על פי מידע מתקבל
- שליחת מידע לשרת
- קבלת מידע מהשרת
- אתחול מסכים שונים
- העלאת מסך שגיאה
- שליחת הודעה
- התחלת שיחה קולית
- התחלת שיתוף מסך
- התחלת שיתוף מצלמה
- יצירת קבוצה
- השמעת קבצי אודיו
- הפעלת קבצי וידאו
- פתיחת מספר קבצים מסוגים שונים

יכולות בצד שרת:

- שם היכולת: שליחת מידע ללקוח

- מהות: פונקציה אשר תשלח מידע למשתמש על פי הכתובת האישית שלו.
 - אוסף יכולות נדרשות: פונקצית שליחת מידע בעזרת sockets, יכולת לזהות את הכתובת של הלקוח.
 - אובייקטים נחוצים: מילון שמכיל בתוכו את שמות הלקוחות ואובייקט המכיל את הכתובות של הלקוחות.
-
- שם היכולת: שמירת מידע בתוך מסד הנתונים
 - מהות: פונקציה אשר תשמור את המידע הנחוץ במקום הנכון בתוך מסד הנתונים.
 - אוסף יכולות נדרשות: פונקציות שיכולת לשמור מידע לתוך מסד נתונים, אפשרות להתחבר אל מסד הנתונים.
 - אובייקטים נחוצים: בסיס נתונים
-
- שם היכולת: קבלת מידע מהלקוח
 - מהות: פונקציה אשר תקבל מידע מהלקוח.
 - אוסף יכולות נדרשות: פונקציה קבלת מידע בעזרת sockets, יכולת לזהות את הכתובת של הלקוח (ממי הגיע ההודעה), פונקציה שתוכל לפענח את ההודעה.
 - אובייקטים נחוצים: פענוח, socket, תקשורת.
-
- שם היכולת: הוצאת מידע ממסד הנתונים
 - מהות: פונקציה אשר מאפשרת למשתמש לאחזר מידע ממסד הנתונים בהתאם לפניותיו.
 - אוסף יכולות נדרשות: יכולת לשאילתת מסד נתונים: יכולת ליצור ולבצע שאילתות SQL כדי לקבל מידע מבסיס הנתונים. אפשרות להתחברות אל מסד הנתונים: יכולת להתחבר ולהתנתק ממסד הנתונים על מנת לבצע פעולות.
 - אובייקטים נחוצים: חיבור למסד הנתונים. פעולות קריאה וכתיבה למסד הנתונים.
-
- שם היכולת: טיפול במידע של מסד הנתונים:

- מהות: פונקציות שמאפשרות עריכה, עדכון, מחיקה וניהול נתונים במסד הנתונים.
- אוסף יכולות נדרשות: יכולת ליצור, לקרוא, לעדכן ולמחוק רשומות ממסד הנתונים. ניהול יחסים: יכולת לנהל יחסים בין טבלאות במסד הנתונים.
- אובייקטים נחוצים: ממשק לבצע פעולות על טבלאות במסד הנתונים. כלי לניהול יחסים במסד הנתונים.

- שם היכולת: ניהול שיחות
- מהות: פונקציות שמאפשרות ניהול וטיפול בשיחות עם הלקוחות.
- אוסף יכולות נדרשות: יצירת שיחה: יכולת ליצור שיחה חדשה עם לקוח ולשוחח איתו. טיפול בשיחה קיימת: יכולת לנהל ולטפל בשיחות קיימות, כולל שליחת הודעות, העברת שיחה, וסיום שיחה.
- אובייקטים נחוצים: מודלים לטיפול בשיחות ובהודעות. פונקציות ושירותים לניהול השיחות.

- שם היכולת: ניהול צלצולים
- מהות: פונקציות שמאפשרות ניהול וטיפול בשיחות טלפון נכנסות ויוצאות.
- אוסף יכולות נדרשות: קבלת צלצול: יכולת לזהות ולטפל בצלצולים נכנסים. שליחת צלצול: יכולת לשלוח צלצולים למשתמשים אחרים.
- אובייקטים נחוצים: ממשק לקבלת צלצולים ולניהולם. פונקציות ושירותים לשליחת צלצולים.

- שם היכולת: ניהול סטרימים
- מהות: פונקציות שמאפשרות ניהול וטיפול בסטרימים של וידאו או אודיו.
- אוסף יכולות נדרשות: יצירת סטרים: יכולת להתחיל סטרים של וידאו או אודיו מהשרת ללקוח או מהלקוח לשרת. ניהול סטרים: יכולת לנהל סטרים פעיל, כולל עצירה, המשך והפסקה של הסטרים. טיפול בשגיאות: יכולת לטפל בשגיאות ובעיות במהלך הסטרים.

אופטימיזציה של סטרים: יכולת לנהל את איכות הסטרים בהתאם לתנאי הרשת והמשאבים הזמינים.

- אובייקטים נחוצים: מודלים לניהול סטרים. פרוטוקולים כמו ממשקים לפענוח וקידוד של מדיה.

יכולות בצד הלקוח:

- שם היכולת: פועל על פי מידע מתקבל:
- מהות: פונקציה אשר מפענחת ומבצעת פעולות בהתאם למידע שמתקבל מהשרת.
- אוסף יכולות נדרשות: פענוח הודעות: יכולת לפענח הודעות והוראות שמתקבלות מהשרת. ביצוע פעולות: יכולת לבצע פעולות שונות על פי המידע המפוענח.
- אובייקטים נחוצים: מילון מיפוי פעולות. מערכת פענוח הודעות.

- שם היכולת: שליחת מידע לשרת:
- מהות: פונקציה אשר שולחת מידע לשרת על פי צורך.
- אוסף יכולות נדרשות: תקשורת עם השרת: שימוש בפרוטוקול TCP/UDP לשליחת המידע. יצירת הודעות: יכולת ליצור ולארז הודעות לשליחה.
- אובייקטים נחוצים: אובייקט תקשורת (socket). מודלים לאריזת מידע (pickle).

- שם היכולת: קבלת מידע מהשרת:
- מהות: פונקציה אשר מקבלת מידע מהשרת.
- אוסף יכולות נדרשות: תקשורת עם השרת: שימוש בפרוטוקול TCP/UDP לקבלת מידע. פענוח הודעות: יכולת לפענח את ההודעות המתקבלות.
- אובייקטים נחוצים: אובייקט תקשורת (socket). מערכת פענוח הודעות.

- שם היכולת: אתחול מסכים שונים:
- מהות: פונקציה אשר מאתחלת מסכים שונים באפליקציה.
- אוסף יכולות נדרשות: יצירת ממשק גרפי: יכולת ליצירת ואתחול מסכים וממשקים. טעינת נתונים: יכולת לטעינת נתונים רלוונטיים למסך המופעל.

- אובייקטים נחוצים: מחלקות למסכים שונים. מערכת ניהול ממשקים גרפיים (GUI).
- שם היכולת: העלאת מסך שגיאה:
- מהות: פונקציה אשר מציגה מסך שגיאה במידה ויש תקלה.
- אוסף יכולות נדרשות: יצירת ממשק גרפי: יכולת ליצירת מסך שגיאה. ניהול שגיאות: יכולת לזהות ולטפל בשגיאות.
- אובייקטים נחוצים: מחלקה למסך שגיאה. מערכת ניהול שגיאות.
- שם היכולת: שליחת הודעה:
- מהות: פונקציה אשר שולחת הודעה למשתמשים אחרים.
- אוסף יכולות נדרשות: יצירת הודעות: יכולת ליצור הודעות לשליחה. תקשורת עם השרת: יכולת לשלוח את ההודעה דרך השרת.
- אובייקטים נחוצים: אובייקט תקשורת (socket). מערכת ניהול הודעות.
- שם היכולת: התחלת שיחה קולית:
- מהות: פונקציה אשר מתחילה שיחה קולית עם משתמש אחר.
- אוסף יכולות נדרשות: תקשורת בזמן אמת: יכולת לנהל שיחה קולית בזמן אמת. קידוד ופענוח אודיו: יכולת לקודד ולפענח אודיו.
- אובייקטים נחוצים: פרוטוקול תקשורת. מערכת קידוד אודיו.
- שם היכולת: התחלת שיתוף מסך:
- מהות: פונקציה אשר מתחילה שיתוף מסך עם משתמש אחר.
- אוסף יכולות נדרשות: לכידת מסך: יכולת ללכוד את מסך המשתמש בזמן אמת. תקשורת בזמן אמת: יכולת לשדר את המסך למשתמשים אחרים.
- אובייקטים נחוצים: מערכת לכידת מסך. פרוטוקול תקשורת.

- שם היכולת: התחלת שיתוף מצלמה:
 - מהות: פונקציה אשר מתחילה שיתוף וידאו ממצלמת המשתמש.
 - אוסף יכולות נדרשות: לכידת וידאו: יכולת ללכוד וידאו מהמצלמה בזמן אמת. תקשורת בזמן אמת: יכולת לשדר את הוידאו למשתמשים אחרים.
 - אובייקטים נחוצים: מערכת לכידת וידאו. פרוטוקול תקשורת.
-
- שם היכולת: יצירת קבוצה:
 - מהות: פונקציה אשר יוצרת קבוצה חדשה למשתמשים.
 - אוסף יכולות נדרשות: ניהול משתמשים: יכולת להוסיף משתמשים לקבוצה. תקשורת עם השרת: יכולת לשמור את הקבוצה בשרת.
 - אובייקטים נחוצים: מערכת ניהול קבוצות. אובייקט תקשורת (socket).
-
- שם היכולת: השמעת קבצי אודיו:
 - מהות: פונקציה אשר משמיעה קבצי אודיו.
 - אוסף יכולות נדרשות: ניהול קבצי אודיו: יכולת לטעון ולהשמיע קבצי אודיו. ממשק גרפי: יכולת ליצור ממשק לניהול השמעת האודיו.
 - אובייקטים נחוצים: נגן אודיו. מערכת ניהול ממשק גרפי.
-
- שם היכולת: הפעלת קבצי וידאו:
 - מהות: פונקציה אשר מפעילה קבצי וידאו.
 - אוסף יכולות נדרשות: ניהול קבצי וידאו: יכולת לטעון ולהפעיל קבצי וידאו. ממשק גרפי: יכולת ליצור ממשק לניהול הפעלת הוידאו.
 - אובייקטים נחוצים: נגן וידאו (OpenCV). מערכת ניהול ממשק גרפי.
-
- שם היכולת: פתיחת מספר קבצים מסוגים שונים:
 - מהות: פונקציה אשר מאפשרת פתיחת קבצים מסוגים שונים.

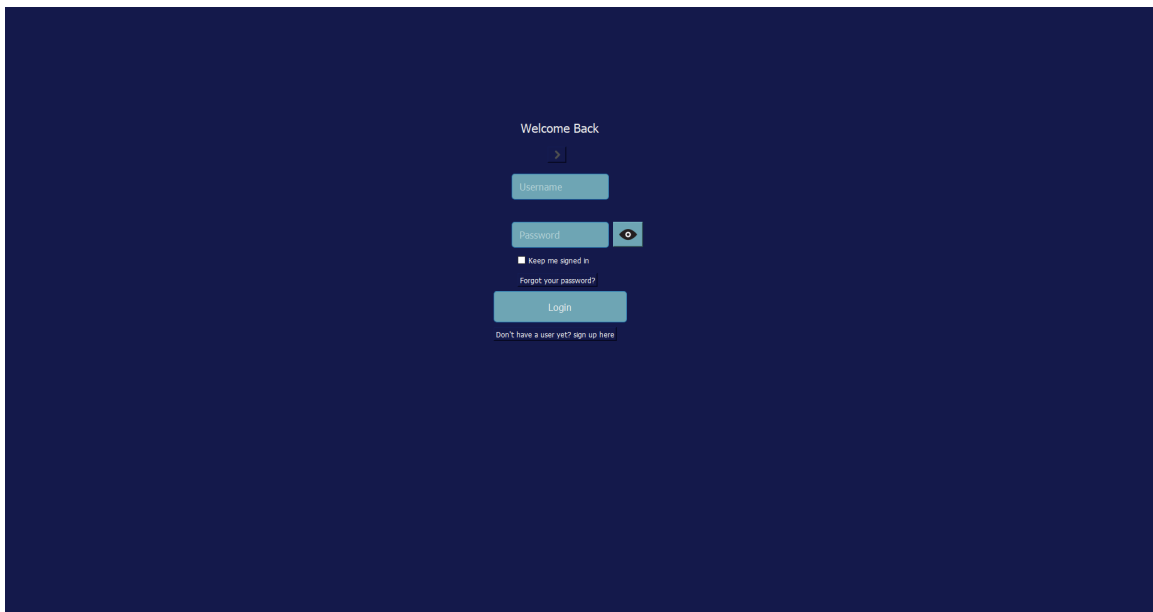
- אוסף יכולות נדרשות: ניהול קבצים: יכולת לטעון קבצים שונים. ממשק גרפי: יכולת ליצור ממשק לניהול פתיחת הקבצים.
- אובייקטים נחוצים: מערכת ניהול קבצים (PyQt). מערכת ניהול ממשק גרפי.

ארכיטקטורה

ממשק גרפי

בפרייקט בסופו של דבר הוא אפליקציה. שבו יכללו כמה דפים עיקריים שהמשתמש יוכל לראות ולהשתמש בהם.

מסך הכניסה/הרישום:



בתמונה ניתן לראות את דף ההתחברות. מדף זה ניתן לעבור לשני דפים נוספים דף ההרשמה ודף השכחתי סיסמא. דף ההתחברות הוא הדף הראשי של האפליקציה לפני כניסת המשתמש ומהווה נקודת בסיס לדפים האחרים. כלומר אם הדפים האחרים רוצים לחזור אחרונה הם חוזרים לדף ההתחברות. מדף ההתחברות אם מקישים את הפרטים הנכונים אפשר להגיע לדף למסך הצ'אט.

Create Account

>

Username

Password

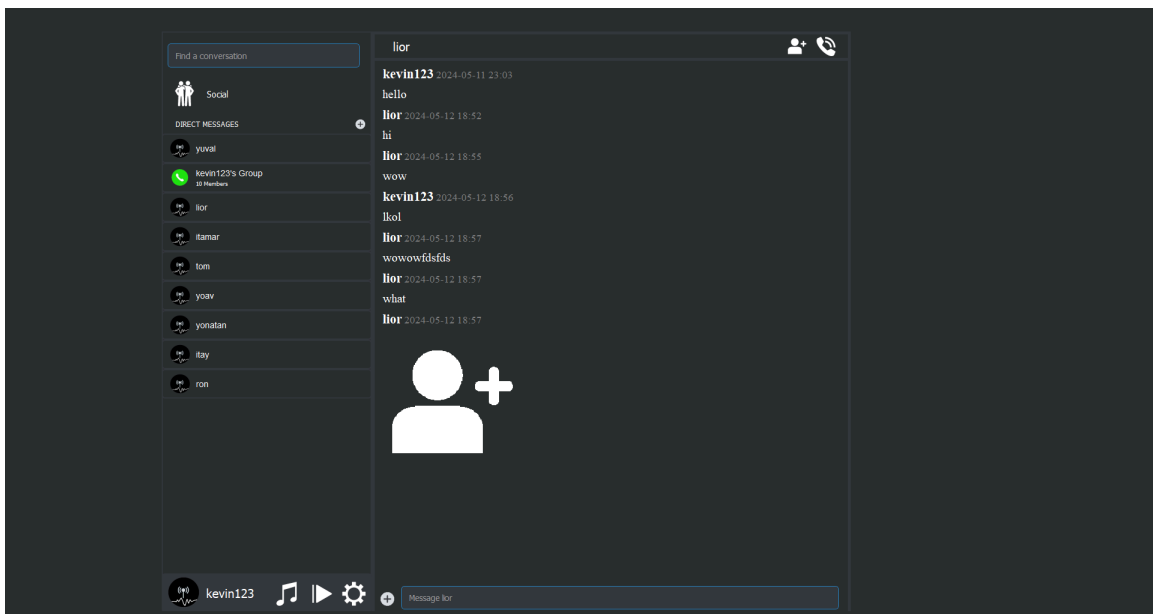
Confirm Password

Email

Submit

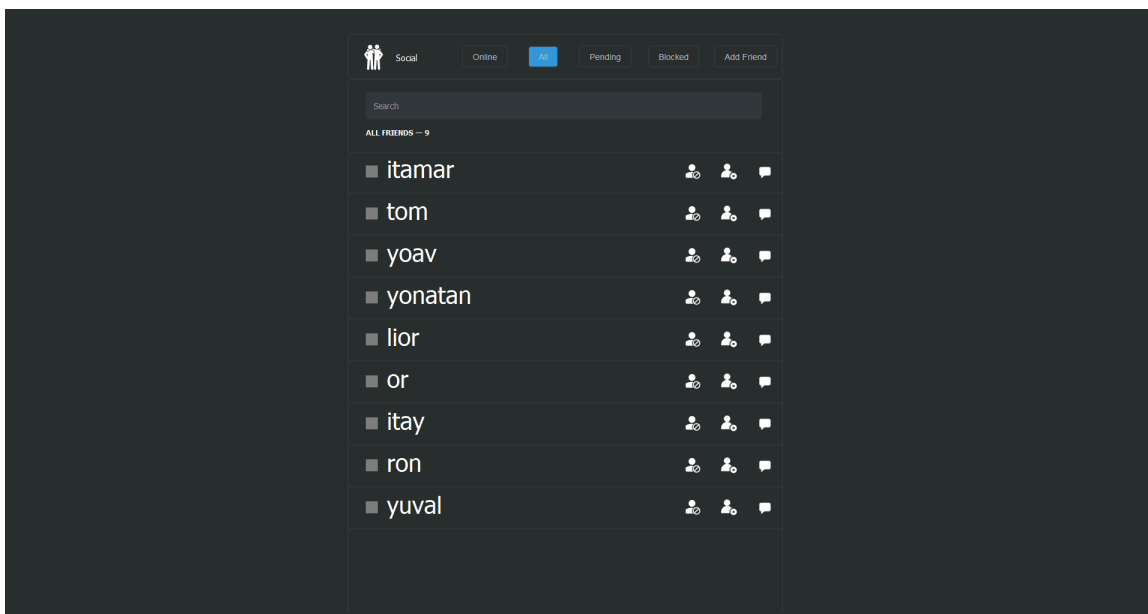
דף ההרשמה הוא מאוד דומה לדף ההתחברות איך מטרתו שונה לגמרי. מדף זה אתה יכול לחזור לדף ההתחברות או למלא את הפרטים שלך ולעבור לדף אישור האימייל שם נשלח קוד שעל ידי הקשתו הלקוח יוצר את המשתמש שלו.

מסך הצ'אט:



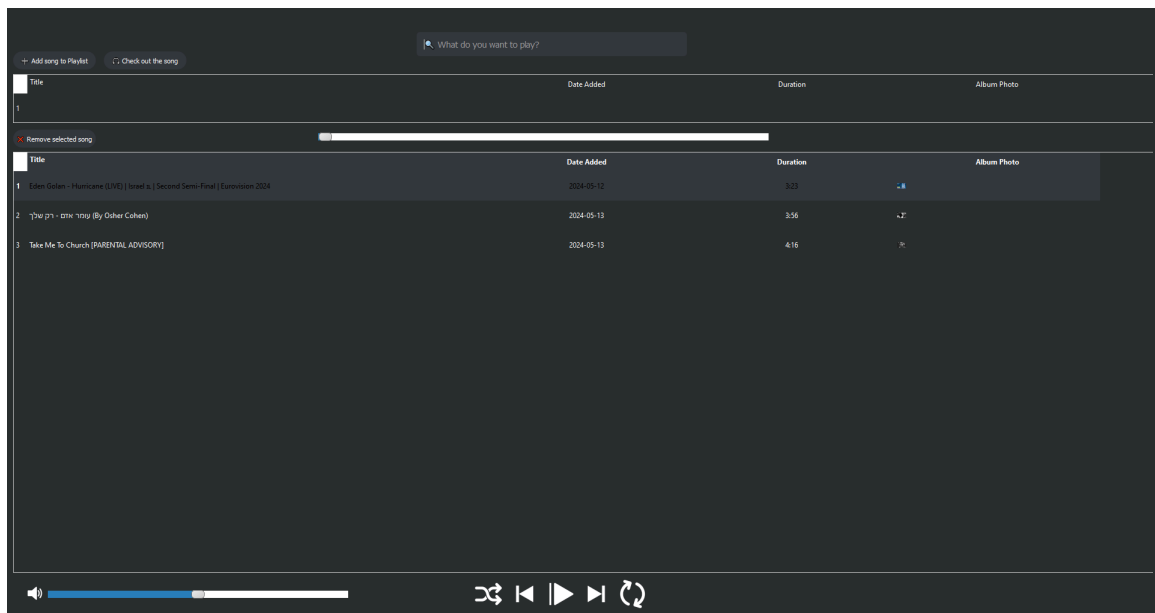
מסך הצ'אט הוא המסך המרכזי של האפליקציה רוב הדברים שניתנים לעשות הם בדף זה. מדף זה אפשר להגיע לשלושה דפים אחרים: דף המוזיקה, דף החברים ודף ההגדרות. בדף זה אתה יכול לשלוח הודעות לצ'אטים הנתונים שלך ואפילו לפתוח שיחות עם משתמשים אחרים.

מסך החברים:



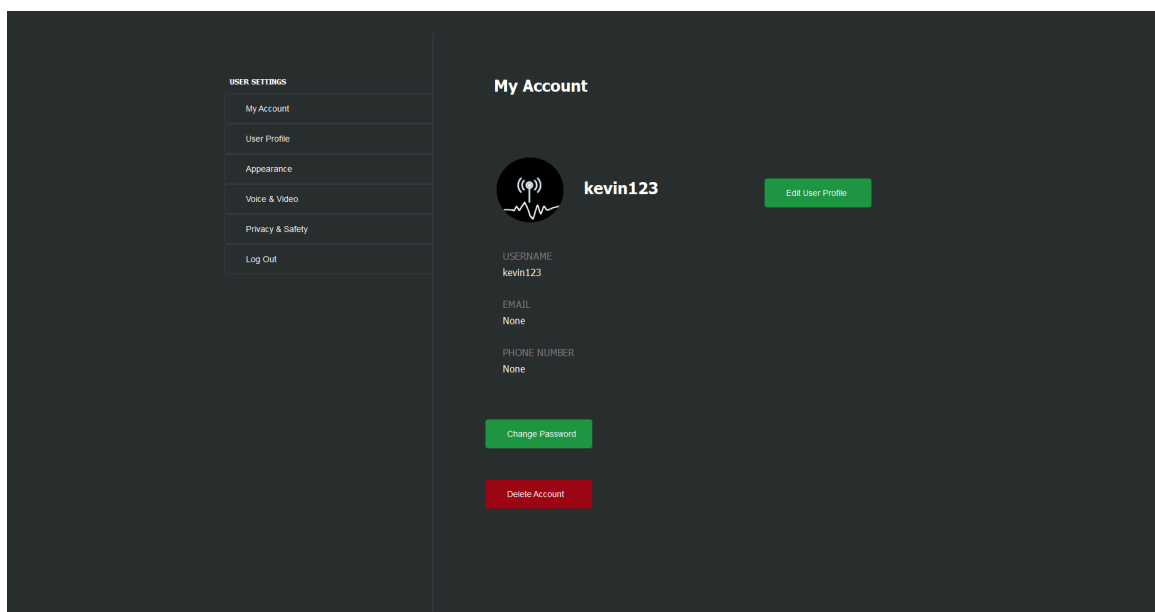
במסך העניינים החברתיים יש את הרשימה של הצעות החברות של המשתמש. Text box בה המשתמש יכול לרשום שם של חבר או משתמש מסוים ובכך לשלוח לו הצעת חברות. בצד השני של המסך יש את רשימת החברים כשלידם יש את הסטטוס שלהם כרגע, האם הם מחוברים למשתמש או לא. אם הם מחוברים הריבוע היה ירוק אם לא הוא אפור. מדף זה אפשר לחזור לדף הצ'אטים.

מסך המוזיקה:



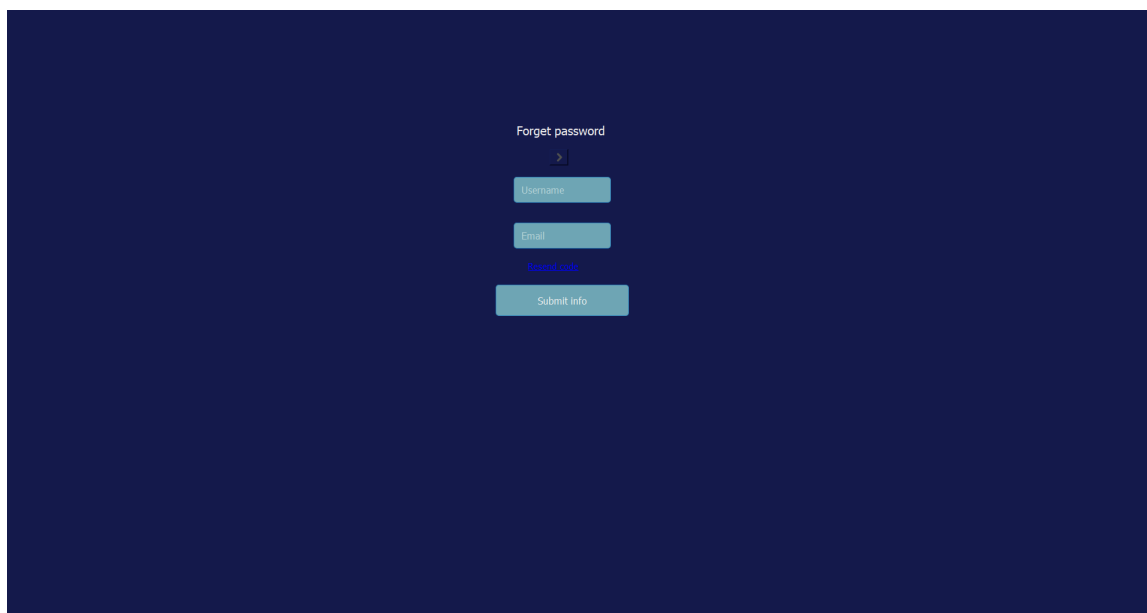
במסך זה יהיה למשתמש גישה לפלייליסט שלו. כלומר ממסך זה המשתמש יוכל לבחור איזה שיר להשמיע שנמצא בפלייליסט שלו, להוסיף שירים ולהעביר ביניהם על פי מידת הצורך של המשתמש. מסך זה נועד במיוחד כדי לשפר את חווית המשתמש ולמנוע שעמום. כאשר המשתמש אינו בשיחה כולשהי עדיין יהיה לו דרך לבדור את עצמו. אפשר לחזור מדף זה לדף הצ'אטים.

מסך ההגדרות:



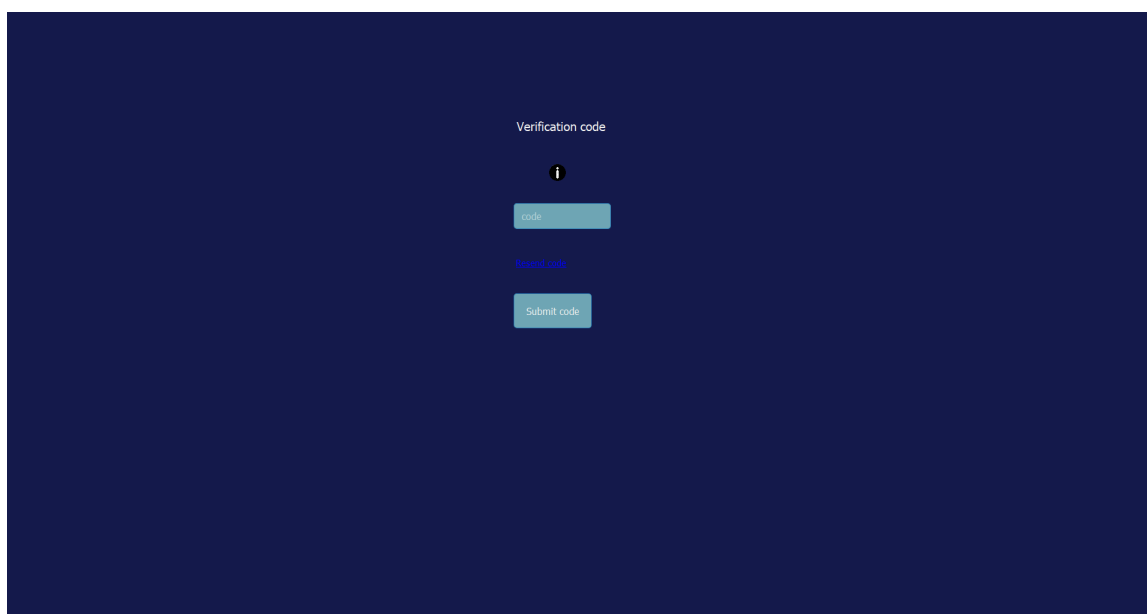
דף זה הוא דף ההגדרות האישיות של המשתמש המשתמש יכול לבחור את ההגדרות על פי העדפותיו האישיות. המשתמש יכול לחזור לדף הצ'אט מדף זה.

מסך שכחתי סיסמא:



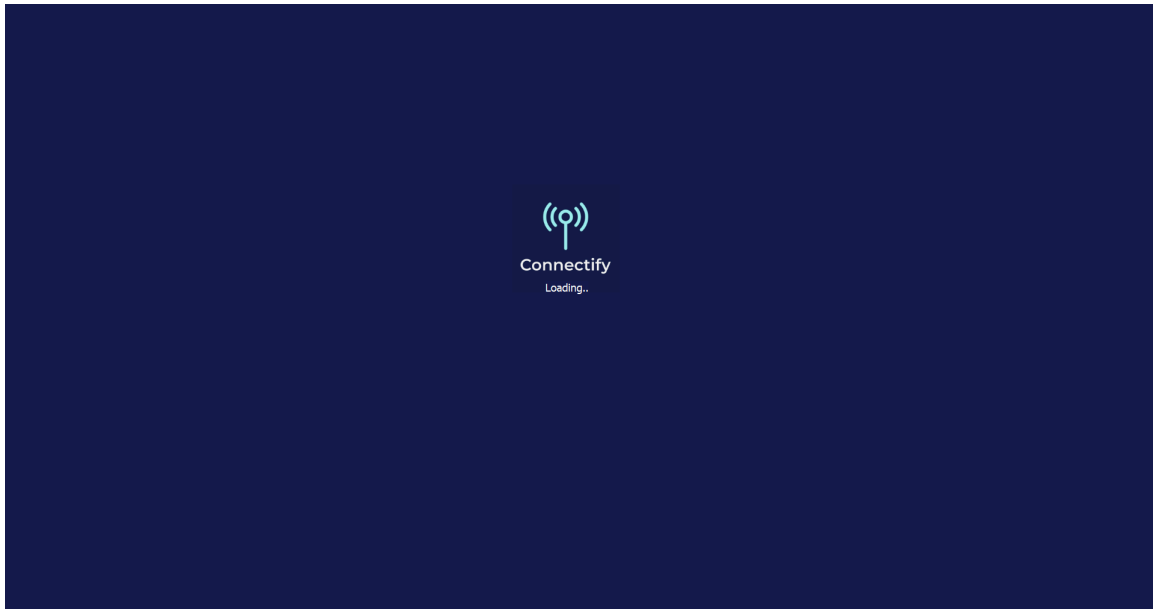
המשתמש צריך למלא פרטים נכונים שקיימים במוסד הנתונים, אם הנתונים נכונים הוא יכול לעבור למסך האישור בו הוא ייקבל קוד באימייל על מנת לשנות את סיסמאתו.

מסך קוד האישור:



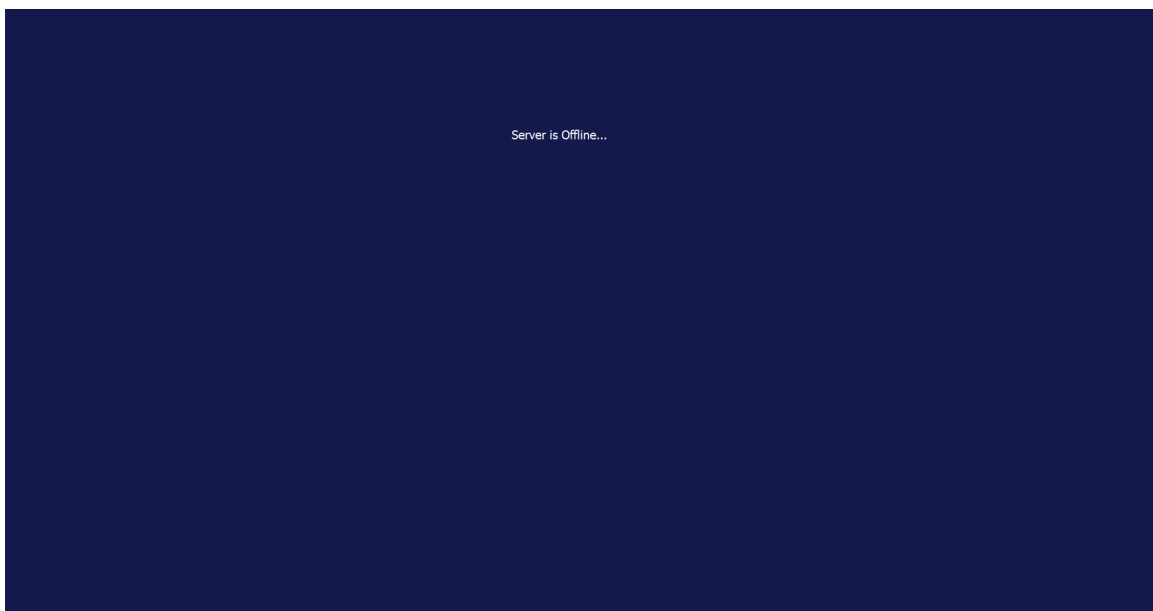
במסך זה המשתמש צריך להקיש את הקוד שהוא קיבל אל האימייל. אם הקוד הוא נכון אז פעולת המשתמש עברה בהצלחה והוא יכול לחזור לדף הראשי אחרי פעולתו.

מסך הטעינה:



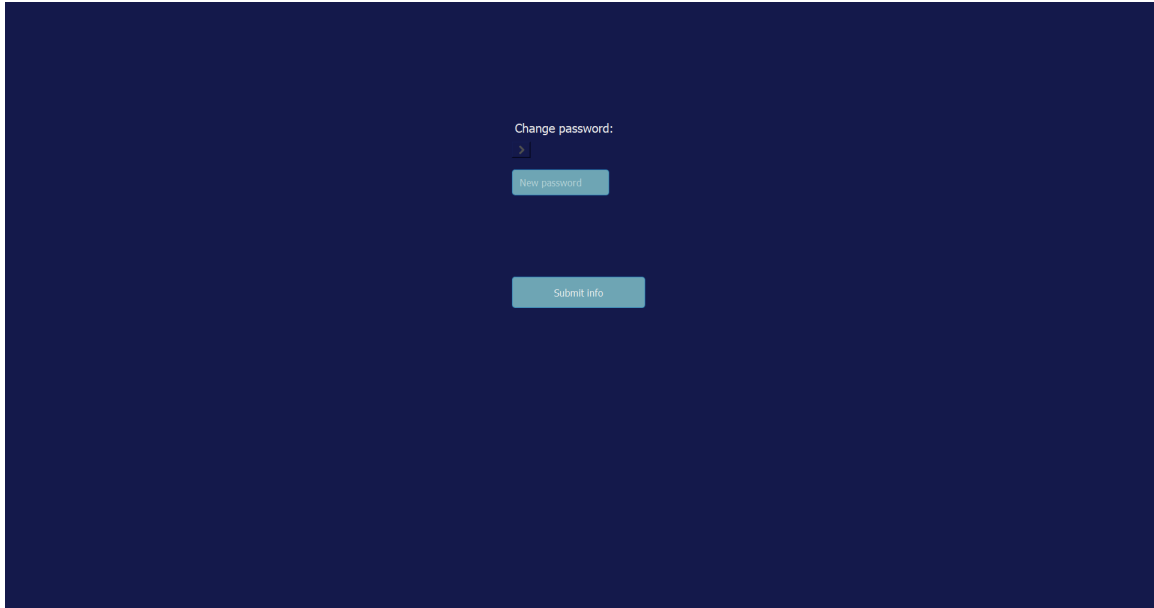
מסך הטעינה פועל כאשר יש צורך בלטעון מידע מסויים. הוא פועל בתחילת האפליקציה כדף ראשוני כדי להחליק את הכניסה אליה. לאחר מכן הוא פועל כטוען, כאשר הלקוח מתחבר הוא מקבל מידע במהלך קבלת המידע הדף הזה עובד לאחר סוף הטעינה המסך עובר לדף הבא.

מסך Error:



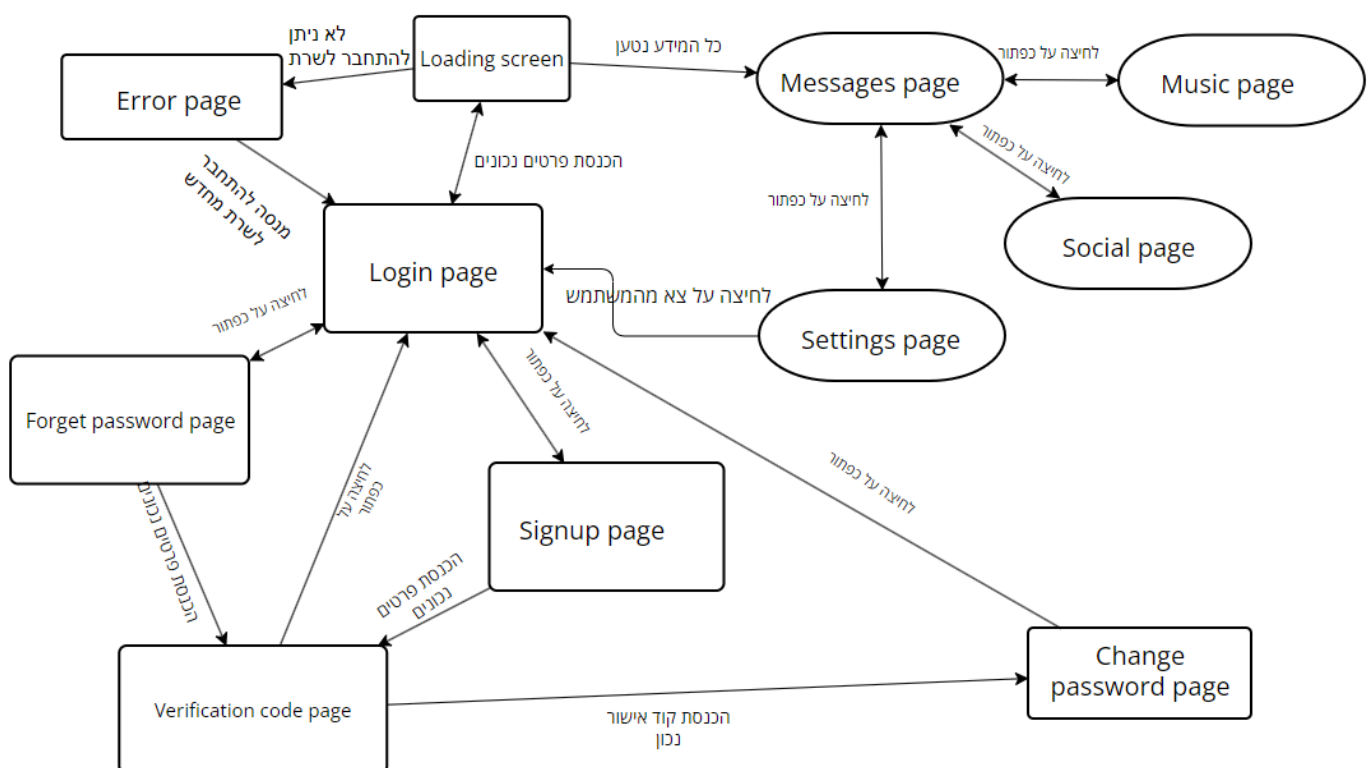
מסך זה נקרא כאשר הלקוח מאבד קשר עם השרת או כאשר שלקוח מנסה להתחבר מתי שהשרת לא פועל.

מסך השנה סיסמא:



לאחר שהקיש פרטים נכונים וקוד נכון שהלקוח קיבל באימייל המשתמש יכול לבחור סיסמא חדשה, אם הסיסמא עומדת בתנאים היא תשלח לשרת ותשונה.

תיאור גלישה המסכים:



סביבת עבודה

סביבת העבודה שאני בחרתי היא python.

סביבת קוד Python מורכב מכלים ותשתית המקלים על תכנות Python, כגון מתורגמן Python, סביבות פיתוח משולבות (IDEs), עורכי טקסט, מנהלי חבילות, סביבות וירטואליות, מערכות בקרת גרסאות, כלי איתור באגים, מסגרות בדיקה, מחוללי תיעוד, כלי בנייה ואריזה. , וכלי עזר לניהול פרויקטים. רכיבים אלה יחד מאפשרים פיתוח, בדיקות ותחזוקה יעילים של קוד, ומבטיחים שפרויקטי Python מאורגנים היטב וניתנים לשיתוף בקלות עם אחרים. בחירת הכלים בסביבת Python יכולה להשתנות בהתאם לדרישות הפרויקט, העדפות אישיות וצרכי שיתוף פעולה בצוות.

Python היא שפת תכנות ברמה גבוהה ורב-תכליתית הידועה בפשטות ובקריאות שלה. הוא מספק סביבת קוד גמישה וחזקה עבור מגוון רחב של יישומים. ניתן להשתמש למטרות כלליות כולל פיתוח אתרים, ניתוח נתונים, למידת מכונה, אוטומציה ועוד.

בנוסף Python מסוגל לבצע קבוצה מגוונת של משימות. אפשר לכתוב סקריפטים לאוטומציה של פעולות שגרתיות, לבנות יישומי אינטרנט, לנתח נתונים ולפתח יישומי שולחן עבודה.

הגיוון וקלות השימוש של Python הפכו אותה לבחירה פופולרית עבור מפתחים מתחילים ומנוסים כאחד, מה שמאפשר להם להתמודד עם מגוון רחב של פרויקטים בתעשיות שונות.

הסיבה העיקרית שבחרתי Python כסביבת העבודה שלי מבחינת בניית הלקוח והשרת היא מכיוון שזהו שפה שאני מאוד מתמצא איתה והיא מאוד נוחה לי. כבר שנים שאני משתמש בה לפתרון בעיות ולוגיות בעזרת קוד, גם משימוש בשפות אחרות אני יכול להגיד ששפה זו באמת הכי נוחה בשבילי ומתאימה כמעט לכל פרוייקט.

pycharm:

PyCharm היא סביבת פיתוח משולבת (IDE) פופולרית ביותר עבור שפת התכנות פייתון. היא פותחה על ידי חברת JetBrains, הידועה בכלי פיתוח איכותיים אחרים כמו IntelliJ IDEA. PyCharm זמינה בשתי מהדורות עיקריות:

מהדורה קהילתית חינמית: זמינה לכל אחד וכוללת מגוון רחב של תכונות עבור פיתוח פייתון. מהדורה מקצועית בתשלום: מציעה תכונות נוספות מתקדמות, כגון תמיכה מדעית נרחבת, פיתוח אינטרנט מתקדם ועוד.

כמה מהיתרונות העיקריים של השימוש ב-PyCharm.:

כלי דיבוג: PyCharm כוללת כלי דיבוג חזקים המאפשרים לך לעקוב אחר ביצוע הקוד שלך, להגדיר נקודות עצירה, לבחון ערכים של משתנים ועוד.

תמיכה במסגרות: PyCharm תומכת במגוון רחב של מסגרות פייתון פופולריות, כגון Django, Flask ו-Web2py.

כלי פרודוקטיביות: PyCharm כוללת מגוון רחב של כלי פרודוקטיביות שיכולים לעזור לך לחסוך זמן ולשפר את זרימת העבודה שלך, כגון השלמת קוד, ניהול קבצים, ניווט פרויקט ועוד.

PyCharm הוא כלי רב עוצמה ורב-תכליתי שיכול לעזור לך לפתח תוכנות פייתון בצורה יעילה ופרודוקטיבית יותר. בין אם אתה מפתח מתחיל או מנוסה, PyCharm מציעה תכונות שיעזרו לך לשפר את זרימת העבודה שלך ולכתוב קוד טוב יותר.

:github

על מנת לשמור על הקוד שלא יהיה מצב שימחק או יאבד השתמשתי ב-github. github היא תוכנה שבה אתה יכול לשמור קבצים של הפרויקט שלך בקלות וביעילות. יש אפשרות לחבר את ה-github שלך ל-pycharm וככה למעשה לעדכן את הגרסה האחרונה שנשמרה ב-github מתי שיש צורך בכך. בעזרת התוכנה אתה יכול לוודא שקטעי קוד לא נעלמים ולהעביר את הפרויקט שלך ממקום למקום בקלות.

מוסד הנתונים:

בפרויקט יהיו כ-6 טבלאות בבסיס הנתונים:

sign_up_table

sign_up_table	
id	INTEGER
username	VARCHAR(255)
password	VARCHAR(255)
email	VARCHAR(255)
salt	VARCHAR(255)
security_token	VARCHAR(255)
chats_list	TEXT
profile_pic_path	VARCHAR(255)
blocked_list	TEXT

מידע על המשתמש לאחר שנרשם לאפליקציה-

- שם משתמש
- סיסמא לאחר שעשו לה hash יחד עם salt + pepper

- האיימיל שאיתו המשתמש התחבר
- salt שאיתו נעשה hash לסיסמא.
- והאסימון אבטחה.
- רשימת הצ'אטים של המשתמש
- רשימת המשתמשים החסומים של המשתמש
- path לתמונת הפרופיל של המשתמש אם קיימת.

messages

messages	
message_id 🔑	INTEGER
sender_id	INTEGER
receiver_id	INTEGER
message_content	TEXT
message_content_path	TEXT
timestamp	TIMESTAMP
type	TEXT
file_name	TEXT

מידע על כל ההודעות שנשלחו באפליקציה-

- שולח ההודעה
- מקבל ההודעה
- תוכן ההודעה
- זמן שליחת ההודעה
- סוג ההודעה (image, video, string)
- path של התוכן. - במקרה שהתוכן הוא לא string file של התוכן נשמר.
- שם file: אם זה file נשמר גם השם שלה.

friends

friends	
id 🔑	integer
user_id	VARCHAR(255)
friend_user_id	VARCHAR(255)
friendship_status	TEXT

טבלה נוספת היא טבלת החברים והבקשות-

- id של המשתמש ששלח את הזמנת החברות
- id של המשתמש שקיבל את ההזמנה
- הסטטוס בין הזמנת החברות (rejected, accepted, pending)

my_groups

my_groups	
group_id 🔑	INTEGER
group_name	VARCHAR(255)
group_manager	VARCHAR(255)
creation_date	timestamp
group_members_list	TEXT
group_image_path	VARCHAR(255)

טבלה נוספת היא טבלת הקבוצות-

- id של הקבוצה
- שם הקבוצה
- מנהל הקבוצה
- תאריך יצירת הקבוצה
- משתתפי הקבוצה

· path של תמונת הקבוצה אם קיימת.

settings_table

settings_table	
user_id	INTEGER
volume	INTEGER
output_device	VARCHAR(255)
input_device	VARCHAR(255)
camera_device_index	INTEGER
font_size	INTEGER
font	VARCHAR(255)
theme_color	VARCHAR(255)
sensor_data	INTEGER
private_account	INTEGER
push_to_talk_bind	VARCHAR(255)
two_factor_auth	INTEGER

טבלת ההגדרות של כל משתמש:

· id של המשתמש שההגדרות שלו

· ווליום של המשתמש

· בחירת output device של המשתמש

· בחירת input device של המשתמש

· בחירת index של המצלמה על המחשב

· גודל font של המשתמש.

· הצבע של האפליקציה אותו בוחר המשתמש

· האם לעשות sensor למידע שנשלח מאנשים שאינם חברים של המשתמש

- כפתור הpush to talk של המשתמש
- האם המשתמש רוצה בlogin שיהיה 2fa.

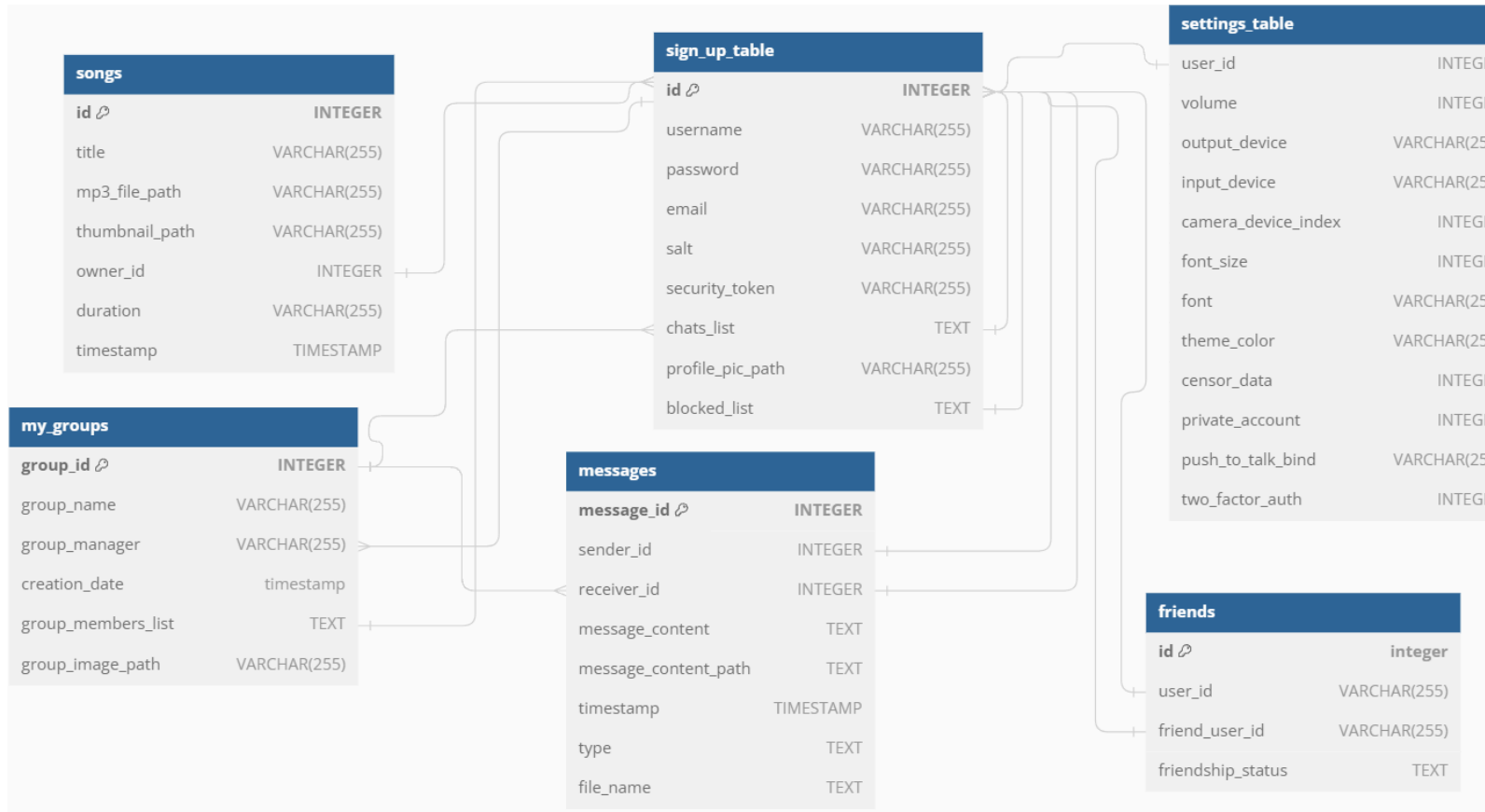
songs

songs	
id 🔗	INTEGER
title	VARCHAR(255)
mp3_file_path	VARCHAR(255)
thumbnail_path	VARCHAR(255)
owner_id	INTEGER
duration	VARCHAR(255)
timestamp	TIMESTAMP

טבלה שמחזיקה את כל השירים של המשתמש:

- כותרת קובץ הmp3
- הpath לקובץ עצמו
- הpath לתמונה של השיר אם קיימת
- הid של הבעלים של הmp3
- הזמן הכולל של הmp3.

קשר בין הטבלאות:



הגדרת פרוטוקול התקשורת

התקשורת במוצר שלי תהיה מובנת על פרוטוקול TCP ועל UDP בחלק מהמקרים.

תחילה בתהליך הקמת הקשר בין השרת לבין הלקוח אני משתמש בפרוטוקול RSA על מנת ליצור מפתח AES משותף לשרת וללקוח כדי שהמידע שיועבר ביניהם יהיה מוצפן.

תהליך החלפת המפתחות:

1. השרת מפרסם את המפתח הציבורי שלו ללקוח
2. הלקוח מכין מפתח AES זמני.

3. הלקוח שולח את מפתח ה-AES מוצפן יחד עם המפתח הציבורי של השרת.
 4. השרת מקבל את מפתח ה-AES מוצפן יחד עם המפתח הציבורי של עצמו.
 5. השרת מפענח בעזרת המפתח הפרטי את ההודעה ומשיג את מפתח ה-AES של הלקוח.
 6. השרת יוצר מפתח AES משל עצמו.
 7. השרת מצפין את מפתח ה-AES שהוא יצר עם המפתח הזמני של הלקוח.
 8. הלקוח מקבל את ההודעה - מפענח את המפתח שהשרת יצר ושני הצדדים מתחילים לתקשר עם מפתח ה-AES שהשרת יצר.
- הסיבה ליצירת מפתח AES נוסף כי בגלל שרצתי שבפרוטוקול השרת בהכרח יקבע את המפתח המסוכם בין הלקוח לשרת. על ידי שליחה של מפתח זמני קודם על ידי הלקוח, זה נותן את הסמכות של בחירת המפתח אל השרת. השרת מחלק מידע אישי של לקוחות בין לקוחות אחרים לכן זה חשוב שבהכרח השרת ידאג שהמפתח מתאים לתקן האבטחה שמתאים לו ומסוכם בין שאר הלקוחות. החילוף הנוסף הזה של מפתחות מוסיף עוד שכבת אבטחה נוספת מכיוון שגם אם מפתח ה-AES הזמני ייחשף המפתח הראשי לא בהכרח ייחשף גם הוא.

פרוטוקול התקשורת שלי מתבסס על מילונים וספריית pickle. אז איך הוא עובד? כל הודעה שנשלחת מהשרת ללקוח או מהלקוח לשרת, השרת או הלקוח מכינים מילון אותו הם שולחים. השליחה לאורך כל הפרויקט עובדת בשליחת bytes כמובן אז על מנת לשלוח את המילון אני משתמש בספריית pickle. ספריית pickle יכולה להפוך כל דבר מרשימה ומילון עד לObject של class bytes מה שמאפשר לי לשלוח מילונים בקלות מאוד.

המפתח המשותף לכל המילונים הנשלחים הוא message_type המפתח הזה למעשה מגדיר את למה ההודעה נצרכת כלומר מה תפקידה. על ידי הבאת message_type ייחודי לכל מקרה הserver והclient יודעים לאילו עוד מפתחות הם מצפים במילון, ויכולים בקלות להבדיל בין הודעות שונות.

TCP:

בשליחת ההודעות דרך tcp socket, אני שולח קודם את הגודל של ההודעה כאשר הגודל עצמו נמצא בגודל קבוע. ככה נעשה receive על מנת לקבל הודעות קודם הוא מקבל את גודל ההודעה ואז הוא מקבל את ההודעה עצמה לפי הגודל שקיבל. זוהי דרך מאוד יעילה לדאוג שלא נאבד מידע ושהודעות מתקבלות בשלימותם.

UDP:

לעמות tcpn udp לא צריך את הפוקנציה הזו של receive by size מכיוון שבpython אתה יכול לקבל הודעה יחידה מקudp תמיד על ידי recvfrom. מה שהוא עושה הוא מקבל את ההודעה שנשלחה אל הסוקט הודעה אחת. כלומר הוא תמיד מקבל הודעה אחת בלי צורך בsize.

udp מאוד יעיל וחשוב לשימוש באפליקציות ומשחקים בהם אתה רוצה להעביר את המידע שלך במהירות כאשר האמינות של המידע פחות חשובה. באפליקציה של כאשר אני עוסק בaudio and video streaming - פרוטוקול udp בא טוב מאוד לשילוב.

לכל רשת יש גודל mtu או Maximum transmission unit במילים פשוטות זה הגודל המקסימלי של פקטה שאתה יכול להעביר בשליחה אחת ברשת שלך. השאלה הנובעת מכאן היא מה קורה כאשר את רוצה להעביר פקטה שגדולה מהmtu של הרשת. אז התשובה לכך היא שצריך לעשות דבר הנקרא fragmentation בפרוטוקול tcp התהליך הזה נעשה אוטומטית ולכן מאוד פשוט לשלוח פקטות שגדולות יותר מהmtu של הרשת. אבל בudp צריך לממש את זה בעצמנו כי זה אינו קיים, לכן לפי הפרוטוקול שלי כאשר אני רוצה לשלוח הודעה שגדולה מהmtu אני עושה לה slice ומחלק אותה לכמה הודעות קטנות יותר על פי הגודל הראשוני של ההודעה. ולכן בצד של הלקוח והסרבר יש מנגנונים שאחראים להרכיב את ההודעה המקורית מחדש ולאפשר מעבר של מידע גדול בעזרת udp.

חושב לציין שmtu כמובן אינו קבוע בין רשת לרשת והפרוטוקול מחלץ את mtu לפי כל רשת כל לקוח.

תהליך ההצפנה:

ברגע שהשרת והלקוח סיכמו על מפתח הוא מתחילים לתקשר רק בתקשורת מוצפנת בעזרת המפתח המוסכם. של צד מכין את ההודעה שלו ולאחר סיום ההכנה מצפין אותה ושולח. לכן כאשר יש מפתח מסוכם כשהשרת או הלקוח מקבלים מידע ההנחה שהוא מוצפן בעזרת המפתח - מנסים לפענח אותו בעזרת המפתח ומעבירים את המידע הלאה.

מימוש הפרויקט

סקירת כל המחלקות המרכיבים את הפרויקט:

מחלקות השרת המרכזי:

מחלקות מיובאים:

<u>שם המודל המיובא</u>	<u>למה מיועד המודל</u>
threading	מודל ליצירת thread בפיתון.
socket	מודל לכתיבת קוד בשילוב socket בפיתון.
json	מודל המאפשר שימוש במחרוזות Json בפיתון.
uuid	מודל המשמש ליצירת מחרוזת ייחודית לכל חיבור חדש לשרת.
random	למה מיועד המודל: מודל המשמש לקבלת מספרים רנדומלים בפיתון, בקוד שלי משמש ליצירת מפתח פרטי רנדומלי בין 0 ל100 בשרת בפרוטוקול דיפי הלמן.
time	למה מיועד המודל: מודל זה נותן אפשרות לחכות מספר מסויים של שניות.
os	מודל המאפשר גישה למערכת הפעלה בפיתון, משמש לגישה לpaths כדי לקבל קובץ מסויים שנשמר על המחשב.
zlib	מודל זה נותן לך אפשרות לעשות zip למידע יש לך אפשרות לעשות לו קיבוץ ולהחזיר אותו למצבו המקורי.
logging	מודל ה-logging ב-Python הוא מערכת גמישה לרישום אירועים ביישומים ובספריות. הוא מאפשר לתעד הודעות, אזהרות ושגיאות בצורה מבוקרת, תוך הגדרת רמות חומרה, פורמטים וסוגי פלט שונים. תוך הכנסת התייעוד לתוך קובץ מסוג log.

מודל הממיר נתונים בינאריים לבסיס 64 ולהפך	base64
מודל שנותן לך גישה לשעה כרגע במחשב הנתון. ניתן לפרמט זמן בדרכים שונות	datetime
מודל הנותן גישה למיקרופונים והרמקולים במחשב.	pyaudio
מודל מאפשר להפוך מידע ל־bytes להחזירו למצבו המקורי	pickle
מודל הנותן גישה למצלמה.	cv2
מאפשר לך ליצור, לנהל ולעבד מערכים רב ממדיים ביעילות.	numpy
מודל הנותן לך גישה לרוזוליצית המסך.	pyautogui
מודל המאפשר חצירת pattern על מנת להשיג ביטוי מ־string.	re
מודל המאפשר לך לערוך תמונות לקחת צילומי מסך.	PIL
מודל המאפשר לך ליצור ולגשת לקבצים זמניים במערכת ההפעלה.	tempfile
מודל המאפשר לעשות פעולות מתמטיות	math
מאפשר לך להפעיל תוכניות ותהליכים אחרים מתוך קוד Python	subprocess
מודל הנותן לך לעשות המון פעולות על strings	string
מודל המאפשר לך יצירת מפתחות באיזה גדול שתרצה.	secrets

מודל המאפשר להעתיק משתנה בפייתון ללא pointer שלו.	copy
מודל המאפשר לשחק עם bytes של file ולהשמיט או לשנות אותם במידת הצורך.	io
מודל המאפשר שימוש במסד נתונים של sqlite המחבר את הפייתון ומסד הנתונים.	sqlite3
מודל המאפשר להפוך מידע אל hash שלו.	hashlib
מודל זה ב-Python מאפשר לך לשלוח דוא"ל דרך שרת SMTP.	smtplib
מודל המקל על שליחה ויצירה של הודעות דוא"ל	email
מאפשר לך להצפין את התקשורת בין האפליקציה שלך לאתר אינטרנט או לשרת, מה שמונע מהאקרים להאזין או לשנות את הנתונים.	ssl
כלי להצפנות סימטריות ואסימטריות. השתמשתי בו ל-AES ו-RSA בקוד.	cryptography
כלי לביצוע בקשות http לשרתי אינטרנט	requests
כלי המאפשר לחפש סרטונים ביוטיוב	youtube_search
מאפשר להוריד סרטונים ביוטיוב כפורמטים שונים	pytube
מאפשר לחתוך ולחבר קטעי וידאו.	moviepy.editor

מחלקות שאני פיתחתי:

שם המחלקה: ServerHandler

תכונות המחלקה:

- calls
- rings
- nets_dict
- udp_addresses_dict
- online_users
- udp_socket
- logger
- udp_socket
- UDPClientHandler_list
- server_mtu

תפקיד ושימוש: מחלקה זו מנהלת את אירועים שקורים בצד של השרת כמו: שיחות, צילולים.
 היא דואג לעוקב אחרי הסטטוס של כל לקוח ולוקחת חלק גדול בעיצוב התקשורת ושמירת נתונים שהסרבר זקוק להם ששימוש בdb אינו יעיל בשבילים.
 פעולות במחלקה:

שם הפונקציה	טענת כניסה	טענת יציאה
update_message_for_users	רשימה של משתמשים, הודעה	שולחת לכל המשתמשים המחוברים את ההודעה שקיבלה.
is_user_online	משתמש	מחזיר האם המשתמש מחובר או לא
add_udp_adress	משתמש, כתובת הudp שלו.	מוסיף את הudp adress למילון שמחזיק את הכתובות.
send_bytes_udp	הודעה, כתובת udp, המשתמש שצריך לקבל	שולח את המידע שקיבל למשתמש בקיבל בudp socket
send_message_dict_udp	מילון של ההודעה, כתובת, שם משתמש	מעביר את ההודעה לpickle על מנת לשלוח את המילון ומעביר אותה לsend_bytes_udp

בודק את גודל ההודעה במידע וזו הודעה גדולה שלא נכנסת בmtu מחלק אותה לכמה הודעות קטנות ומעביר אותם לשליחה.	שם השולח, שם המקבל, המידע, סוג המידע, צורת frame או זה frame.	send_large_udp_data
פונקציה שבודקת את גודל mtu של השרת	כתובת זמנית של לקוח	check_max_packet_size_udp
שולח את המידע של הקבוצה החדשה לכל המשתתפים המחוברים של הקבוצה.	id של קבוצה	send_new_group_to_members
שולח את המילון המעודכן של קבוצה שהשתנתה לכל המשתתפים המחוברים.	id של קבוצה	update_group_dict_for_members
משיג את המידע הבסיסי שהמתמש צריך שיהיה לו ברגע הכניסה ושולח לו	שם משתמש	send_user_needed_info
שולח את הפרופילים שרולוונטים למשתמש בעת כניסתו.	שם משתמש, תמונה שמקודדת כbase64	update_profiles_list_for_everyone_by_user
שולח את הפרופיל החדש והמעודכן למשתמש הנתון.	שם משתמש לשלוח לו, תמונה שמקודדת כbase64, שם המשתמש של בעל הפרופיל	send_new_profile_of_user
שולח למשתמש את רשימה של המילונים הרלוונטים של פרופילים של משתמשים אחרים.	שם משתמש	send_profile_list_of_dicts_to_user
מחזיר רשימה של מילונים של כל השיחות של הלקוח.	שם לקוח	get_list_of_calls_for_user
מעדכן את רשימה המשתמשים שמחוברים למשתמש.	שם הלקוח	update_online_list_for_users_friends

user_online	שם הלקוח, netn של הלקוח (מסוג class client_net)	מעדכן שהמשתמש מחובר ומוסיף את הנתונים שלו למקום הנכון- בנוסף מעדכן מה שצריך ללקוחות אחרים.
user_offline	שם הלקוח, netn של הלקוח (מסוג class client_net)	מעדכן שהמשתמש לא מחובר ומוריד את הנתונים שלו למקום הנכון - בנוסף מעדכן מה שצריך ללקוחות אחרים.
add_net	שם הלקוח, netn של הלקוח (מסוג class client_net)	מוסיף את netn של הלקוח למילון בנוסף מעדכן עבור שאר המחלוקות שנוספה רשת של לקוח.
update_nets_for_child_class	אין	מעדכן את הרשימה של nets עבור מחלוקות שמסמכות על זה - ובודק את ההשפעות של העדכון ופועל על פי זה.
get_net_by_name	שם	מחזיר את net object מהמילון לפי השם המבוקש אם קיים.
remove_net_by_name	שם	מוריד את השם מהמילון של nets ודואג לעדכונים שיש בהם צורך.
add_call	שיחה (מסוג Call שהיא מחלקה)	מוסיף את השיחה לרשימת השיחות.
is_user_in_call	שם של משתמש	בודק אם המשתנה בשיחה אם כן מחזיר True אם לא False.
remove_user_from_call	שם הלקוח	מוציא את המשתמש מהאנשים שנמצאים בשיחה שהוא כרגע נמצא אם נמצא.
are_users_in_call	רשימה של לקוחות	מחזיר True אם כל הלקוחות בתוך שיחה יחד False אם לא

שם הלקוח	get_call_members_with_user	אם הלקוח בשיחה יחזיר את רשימה הלקוחות שאיתו בשיחה.
תעודת זהות של קבוצה, רשימה של לקוחות	create_call_and_add	יוצר object מסוג Call על ידי הפרמטרים שקיבל ומוסיף את השיחה לנתונים הקיימים.
עצם מסוג Ring	add_ring	מוסיף אותו לרשימה של הצלולים
עצם מסוג Ring	remove_ring	מסיר אותו לרשימה של הצלולים
לקוח שמתקשר, הלקוח אליו הוא מתקשר	create ring	יוצר עצם מסוג Ring לפי הפרמטרים ועושה עדכונים חשובים.
תעודת זהות של קבוצה	is_group_call_exists_by_id	אם קיימת שיחה לפי התעודת זהות מחזיר True אחרת False
שם הלקוח, מידע האודיו	send_vc_data_to_call	מוסיף את המידע אל התור של מידע האודיו של בה נמצא הלקוח.
מידע שיתוף המסך, צורת המידע, המשתמש ששלח, סוג הזרם	send_share_screen_data_to_call	מוסיף את המידע לתור של מידע השיתוף מסך או מצלמה.
תעודת הזהות של השיחה, שם הלקוח	add_user_to_group_by_call_id	אם קיימת שיחה אם תעודת הזהות זהו המשתמש יוסף לשיחה.
הלקוח שהתקשר, הלקוח המנתק	reject_ring_by_ringer	מגיע לעצם Ring ומידע כי הלקוח ניתק
הלקוח שהתקשר, הלקוח העונה	accept_ring_by_ringer	מגיע לעצם Ring ומידע כי הלקוח ענה
הלקוח שהתקשר	cancel_ring_by_the_ringer	מוחק את עצם Ring ומבטל את הצלול.

mute_or_umute_self_user	שם הלקוח	משתיק את הלקוח אם הוא לא מושתק ומבטל אם כן, בנוסף מעדכן את זה בשיחה עצמה.
deafen_or_undeafen_self_user	שם הלקוח	מחריש ומשתיק את הלקוח אם הוא לא ואם הוא כן מבטל את זה בשיחה.
get_ring_id_by_possible_ringers	רשימה של לקוחות	אם אחד הלקוחות נמצא כמי התקשר לאחד הצלולים הקיימים הפונקציה תחזיר את התעודת זהות של Ring המקורי
cancel_ring_by_id	תעודת זהות של Ring	עוצר ומוחק את Ring שמתאים לפרמטר
create_video_stream_for_user_call	שם הלקוח, סוג הזרם	יוצר זרם בשיחה שהלקוח נמצא על פי הפרמטרים
close_video_stream_for_user_call	שם הלקוח, סוג הזרם	סוגר זרם בשיחה שהלקוח נמצא על פי הפרמטרים
add_spectator_to_call_stream	שם הצופה, שם הסטרימר, סוג הסטרים	מוסיף לStream עצמו את הצופה
remove_spectator_from_call_stream	שם הצופה	מסיר לStream עצמו את הצופה
handle_udp_fragment	מקבל fragment של הודעת udp, הכתובת של הלקוח ששלך	מוסיף את ההודעה לעצם של הכתובת שיטפל באותה שקיבל.
create_and_add_udp_handler_object	שם הלקוח, הכתובת udp שלו, הכתובת tcp שלו.	יוצר עצם מסוג UDPCliientHandler ומחזיק אותו ברשימה של המחלקה
get_aes_key_by_username	שם הלקוח	מחזיר את מפתח aes איתו הלקוח מתקשר עם השרת.

שם המחלקה: UDPClientHandler

תכונות המחלקה:

- logger
- udp_address
- tcp_address
- ServerHandler_object
- client_username
- aes_key
- lost_packets
- gotten_packets
- vc_data
- share_screen_data
- share_camera_data

תפקיד ושימוש: מחלקה המשמשת לטיפול בהודעות קד שמתקבלות מן הלקוח על הסרבר. למחלקה יש עצם על כל כתובת קד ומטפלת בהודעות על פי סוגם. היא יכולה להרכיב הודעות הנשלחות בצורה של מספר הודעות קטנות, מרכיבה אותם מנהלת אותם ושולחת אותם למקום הנכון.

שם הפונקציה	טענת כניסה	טענת יציאה
decrypt_data	המידע עצמו	מנסה לפענח את המידע במידה ומצליח מחזיר אותו אם לא מחזיר None

handle_udp_message	חלק מהמידע או המידע	במידע וזה המידע המלא מעביר אותו הלאה, אם חלק מהמידע מרכיב אותו וכאשר הוא שלם שולח אותו - יש טיפול באיבוד פקטות.
handle_data_fragment	רשימה המידע הקיים, המידע החתוך, האם הודעה ראשונה, האם אחרונה, סוג ההודעה, צורת ה frame או frame	כאשר המידע לא שלם היא מוסיפה את המידע לרשימה הנכונה, אם שלם הוא מורכב ומובא הלאה, אם בניסיון ההרכבה יש תקלה מוגדר כאיבוד פקטה ועושה reset למשתנים.

שם המחלקה: VideoStream

תכונות המחלקה:

- call_parent
- comms_parent
- logger
- stream_id
- streamer
- stream_type
- is_group_stream
- spectators
- data_collection
- stop_thread
- thread

תפקיד ושימוש: מחלקה זו היא עצם של Stream קיים, היא מנהלת את המידע הקשור ל Stream ומחזיקה בו. מאפשרת לנהל את השיתוף מסך או מצלמה בצורה חכמה ויעילה.

שם הפונקציה	טענת כניסה	טענת יציאה
remove_spectator	שם הצופה	מוציאה את המידע הקשור אל הצופה מהנתונים ומורידה אותו מרשימת הצופים
add_spectator	שם הצופה	מוסיפה את המידע הקשור אל הצופה אל הנתונים ומוסיפה אותו אל מרשימת הצופים
process_share_screen_data	אין	הthread האחראי על שליחת המידע של הסטריים אל הלקוחות. פועל באופן קבוע עד סגירת הstream
stop_processing	אין	עוזר את הthread האחראי על שליחת מידע הstream קורא בעת סגירת הstream או כשאין צופים.
send_share_screen_data_to_everyone_but_user	המידע של שיתוף המסך או המצלמה, שם המשתמש ששולח, הצורה של הframe	שולח את המידע לכל הצופים לפי הפרמטרים
end_stream	אין	סוגר את הStream ומוחק אותו
adding_share_screen_data_to_user_call_thread_queue	שם המשתמש המוסיף, המידע של השיתוף עצמו, הצורה של הframe	מוסיף את המידע אל התור על מנת שהthread יוכל לשלוח אותו לצופים

שם המחלקה: Ring

תכונות המחלקה:

- logger
- parent
- ring_time
- nets_dict
- ringer
- ring_id
- is_group_ring
- ringing_to
- initiated_time
- ringers_nets
- ringed_to
- already_ringed_to
- ring_thread_flag
- accepted_rings
- rejected_rings
- is_ringer_stopped_call

תפקיד ושימוש: המחלקה היא למעשה עצם של Ring מחזיקה את כל הנתונים הקשורים אל הצלצול שנוצר משיחה כולשהי: מי התקשר, מי ענה מי לא ענה עדיין ועוד. המטרה של המחלקה היא לנהל בקלות את הצלצולים כדי לעדכן לשלוח אותם להיפטר מהן כאשר מיצו את מטרם.

שם הפונקציה	טענת כניסה	טענת יציאה

מוסיף את הלקוח לאנשים שנתקו את הצלול	שם הלקוח שמנתק	rejected_ring
מוסיף את הלקוח לאנשים שענו לצלול	שם הלקוח שענה	accepted_ring
מדובר בthread שרץ מתחילת הצלול ובודק האם הגיע הזמן של הצלול להסתיים - אם נגמר הcooldown או כי כולם ענו או ניתקו.	אין	process_call_and_send_response
כאשר נגמר הזמן של הצלול פונקציה זו דואגת להפסיק אותו אצל כל המשתמשים שלא הגיבו.	אין	stop_ring_for_unanswered_users
מפסיק את הצלול בשביל הלקוח שהתקשר אחרי שניתק.	אין	stop_ring_for_ringer
הלקוח שהתקשר החליט להפסיק את הצלול פני שמישהו ענה - לכן הצלול נעזר ונמחק.	אין	cancel_ring_for_all
מאתחל את הnets של המשתמשים שמצללים אליהם.	אין	gets_ringing_nets_from_dict
מצלל לכל הלקוחות שמחוברים על השרת.	אין	ring_to_everyone_online
מעדכן את הnets של הלקוחות שמתקשרים אליהם. קורה כאשר לקוח נכנס באמצע צלול.	אין	update_ring_nets

שולח צלצול לכל המשתמשים שנמצאים בלקוחות שאמורים לקבל אך לא קיבלו מסיבה מסויימת.	אין	ring_to_users_who_didnt_get_a_ring
אם הלקוח שהפונקציה קיבלה הוא הלקוח שיזם את הצלצול מחזיר True אחרת False.	לקוח שאולי התקשר	is_ring_by_ringer

שם המחלקה: SignUpPage

תכונות המחלקה:

- page_controller_object
- password_not_match_label
- email_required_field
- username_required_field
- password_required_field
- confirm_password_required_field
- invalid_email
- password_too_short
- username_already_used

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. זהו דף ההירשמות של הלקוח בו הוא יכול למלא את הפרטים שלו כדי להירשם.

שם הפונקציה	טענת כניסה	טענת יציאה
-------------	------------	------------

מחביא את כל התוויות שנעודו להצביע למשתמש כי יש בעיה כולשהי	אין	hide_every_error_label
יוצר תווית עם טקסט במיקום הנתון על הדף	טקסט, מיקום	create_label
מוסיף כפתורים ותוויות התחלתיים אל המסך.	אין	init_ui
כאשר המשתמש מסיים להירשם ולוחץ על כפתור ההירשמות הפונקציה הזו נקראת. היא בודקת את input אם הוא תקין זה עובר הלאה תוך תקשורת עם השרת.	שם המשתמש, הסיסמא, הסיסמא בפעם השנייה, אימייל	submit_form
כאשר כפתור החזרה מלחץ המשתמש חוזר לדף הראשי שהוא דף הכניסה.	אין	return_button_pressed

שם המחלקה: LoginPage

תכונות המחלקה:

- page_controller_object
- visibility_password_button
- show_password_icon
- hide_password_icon
- current_icon
- password
- username_entry

- remember_me_status
- incorrect_label
- user_is_logged_in

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. זהו דף הכניסה של הלקוח בו הוא יכול למלא את הפרטים שלו כדי להיכנס למשתמש הקיים שלו או לעבור לדפים אחרים אותם הוא צריך.

שם הפונקציה	טענת כניסה	טענת יציאה
init_ui	אין	מוסיף כפתורים ותוויות התחלתיים אל המסך.
on_checkbox_change	מצב	כאשר התיבון סימון במצב 2 רואים את ה V כאשר המצב שונה לא רואים. התיבה מסמנת האם המשתמש רוצה שהפרטים שלו ישמרו על המחשב אם כן זה ישמור אותם לפעם הבאה שהוא יכנס.
submit_form	אין	לוקח את הסיסמא והשם משתמש שהלקוח הכניס ושולח לשרת אם הם תקינים. לפי התשובה של השרת מחליט איך לפעול אחר כך.
forgot_password_clicked	אין	מחובר לכפתור השכחתי סיסמא. אם הלקוח שכח את הסיסמא הוא יכול לשנות אותה בדף אחר שהוא נכנס על ידי כפתור זה.

פונקציה זו מעבירה את הלקוח לדף הרשמות בו הוא יכול להירשם.	אין	move_to_sign_up_page
פונקציה זו מראה ומחביאה את הסיסמא על פי רצון הלקוח. אם הוא רוצה לראות אותה הוא יכול וכך גם להסתיר	אין	show_password_button_pressed

שם המחלקה: SplashScreen

תכונות המחלקה:

- page_controller_object
- dot_count
- loading_timer
- elapsed_time

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. זהו דף הטעינה הכללי של האפליקציה. הדף קורה כאשר הלקוח מתחבר ומחכה למידע, המידע נטען ובשביל למנות תקיעות אצל הלקוח הדף פועל בזמן זה. דף זה הוא גם הדף הראשון שהלקוח רואה כאשר הוא מפעיל את האפליקציה.

שם הפונקציה	טענת כניסה	טענת יציאה
init_ui	אין	מוסיף תוויות התחלתיות אל המסך ואת האייקון של האפליקציה.

מתחיל טיימר של כמה שניות להתחלה הראשונית של הדף. (בתחילת האפליקציה אין באמת מה לטעון אז זה טעינה בשביל העיצוב. לאחר מכן יש טעינה שקוראת כאשר הלקוח מחכה למידע מהשרת)	אין	start_timer
סוגר את הדף הזמני הזה ועובר אל הדף הראשי. קורה במהלך הטעינה של הדף הראשי.	אין	close_page_open_main_page
כדי להראות טעינה מופיעות נקודות על המסך. הפונקציה הזו דואגת לשנות ולהראות אותם.	אין	update_loading_dots

שם המחלקה: ForgetPasswordPage

תכונות המחלקה:

- page_controller_object
- username
- email

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. דף זה הוא דף השכחתי סיסמא, בדף זה עם הפרטים הנכונים הלקוח יכול לעבור לדף אחר שישנה את סיסמתו. בדף זה הלקוח מכניס שם משתמש עם אימייל תקין ובכך יכול לעבור לדף הבא לשם שינוי הסיסמא.

שם הפונקציה	טענת כניסה	טענת יציאה
init_ui	אין	מוסיף תוויות התחלתיות אל המסך וגם כפתורים.
create_label	טקסט, מיקום	יוצר תוויות במיקום הנתון עם הטקסט הנתון בדף זה.
return_button_pressed	אין	חוזר לדף הראשי.
submit_form	אין	אם המידע שהלקוח הכניס תקין. הוא נשלח לסרבר ועל פי התשובה מעביר אותו דף. עם תשובה חיובית עובר לדף האישור אם שלילי יכול או להישאר בדף או לחזור לדף login.
resend_code_clicked	אין	במידה והלקוח לא קיבל את הקוד יכול ללחוץ על כפתור שישלח לו את הקוד שוב.

שם המחלקה: VerificationCodePage

תכונות המחלקה:

- page_controller_object
- info_label
- code
- successfully_signed_up

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. דף זה הוא דף הקוד אישור. בדף זה מתקבלים קודים אל האימייל הלקוח הכניס וכאן הלקוח מכניס את הקודים עם סנכרון עם השרת וממשיך לפי התגובה של השרת. הגעה אל דף זה מגיעה משלוש אפשרויות: שכחתי סיסמא, הירשמות, 2FA. בכל מצב נשלח הודעה הולמת משני הכיוונים איך הGUI עובד אותו דבר.

שם הפונקציה	טענת כניסה	טענת יציאה
init_ui	אין	מוסיף תוויות התחלתיות אל המסך וגם כפתורים.
resend_code_clicked	אין	במידה והלקוח לא קיבל את הקוד יכול ללחוץ על כפתור שישלח לו את הקוד שוב.
eventFilter	אובייקט, event	הפונקציה מטפלת במצב בו הלקוח עובר מעל התווית של המידע, במקרה זה התווית נראית לעין כדי שהלקוח יוכל לקרוא אותה.
return_button_pressed	אין	חזרה לדף הראשוני שהוא דף login
submit_form	אין	אם אורך הקוד תקין, שולח את הקוד אל השרת ומחכה לתשובה כדי לדעת מה לעשות.
create_label	טקסט, מיקום	יושר את התווית במיקום שהתקבל עם הטקסט שהתקבל

שם המחלקה: ChangePasswordPage

תכונות המחלקה:

- page_controller_object
- change_password_label
- new_password
- status
- image_button
- too_short
- was_password_changed
- password_already_changed

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. בדף זה ניתן למשתמש להכניס סיסמא חדשה אליה הוא רוצה להחליף. במידה והסיסמא תקינה היא נשלחת לסרבר ומתחלפת והלקוח יכול לחזור לתפריט הראשית.

שם הפונקציה	טענת כניסה	טענת יציאה
init_ui	אין	מוסיף תוויות התחלתיות אל המסך וגם כפתורים.
resend_code_clicked	אין	במידה והלקוח לא קיבל את הקוד יכול ללחוץ על כפתור שישלח לו את הקוד שוב.

submit_form	אין	אם הסיסמא שהוכנסה תקינה. הפונקציה תשלח את הסיסמא החדשה אל הסרבר שתשנה אותה והלקוח יוכל לחזור אל דף הlogin.
create_label	טקסט, מיקום	יושר את התווית במיקום שהתקבל עם הטקסט שהתקבל
return_button_pressed	אין	חזרה לדף הראשוני שהוא דף login

שם המחלקה: ServerIsDownPage

תכונות המחלקה:

- page_controller_object
- server_is_offline

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. דף זה מתריע את המשתמש כי הוא
איבד חיבור עם השרת. דף זה מוראה גם כאשר יש ניסיון ראשוני של התחברות כושלת אל
השרת.

שם הפונקציה	טענת כניסה	טענת יציאה
reconnect_to_server	אין	בעת לחיצת כפתור הלקוח יכול לנסות להתחבר מחדש אל השרת.

שם המחלקה: PageController

תכונות המחלקה:

- screen_width
- screen_height
- app
- receive_thread_after_login
- is_logged_in
- is_waiting_for_2fa_code
- splash_page
- sign_up_page
- forget_password_page
- login_page
- main_page
- change_password_page
- verification_code_page
- current_page
- server_is_down_page
- network

תפקיד ושימוש: המחלקה אחראית על הדפים הראשיים שקיימים. היא מחזיקה בה את יצירת הדפים העדכון שלהם והשינוי שלהם. מחלקה זו היא למעשה הבסיס של כל ה-GUI ואפליקציית הלקוח היא אחראית על הכל מסוגלת לסגור הכל במידת הצורך.

שם הפונקציה	טענת כניסה	טענת יציאה

מתחילה את הthread שמקבל את ההודעות מהשרת לאחר הlogin	אין	start_receive_thread_after_login
סוגר את הלקוח לחלוטין	אין	quit_application
מודיע לשרת כי הלקוח מתנתק ומאתחל את כל המידע והדפים הקיימים.	אין	log_out
מנקה את כל הדפים	אין	clear_all_pages
סוגר את כל הדפים	אין	close_all_pages
מחביא את כל הדפים מהלקוח	אין	hide_all_pages
משנה את הדף לדף ההתחברות	אין	change_to_login_page
משנה את הדף לדף הרשמות	אין	change_to_sign_up_page
משנה את הדף לשנות סיסמא	אין	change_to_change_password_page
משנה את הדף האישור	אין	change_to_verification_code_page
משנה את הדף הראשי	אין	change_to_main_page
משנה את הדף לדף השכחתי סיסמא	אין	change_to_forget_password_page
משנה את הדף לדף הטעינה	אין	change_to_splash_page

שם הדף	change_page	שמה לדף מסוים לפי השם שלו
אין	thread_recv_messages	thread ההודעות לאחר ההתחברות, מקבל הודעות ומטפל בהם.

שם המחלקה: VideoClient

תכונות המחלקה:

- main_page
- central_widget
- image_label

תפקיד ושימוש: חלק ממחלקות ה-GUI, המחלקה אחראית להציג סטריים מסוג מצלמה או שיתוף מסך. הדף מקבל frame ומציג אותו על הדף המשתמש רואה את השיתוף על מסך מלא בדף הזה.

שם הפונקציה	טענת כניסה	טענת יציאה
init_ui	אין	מוסיף תוויות התחלתיות אל המסך וגם כפתורים.
display_frame	פריים	מציג את הפריים שהתקבל על המסך
keyPressEvent	אירוע (לחיצה על כפתור)	פונקציה בנויה ב-pyqt5 שנקראת על כל לחיצת כפתור. עבור כפתורים מסויימים עושה פעולות שונות (יציאה מהצפייה בשיתוף)

שם המחלקה: MainPage

תכונות המחלקה:

- page_controller_object
- vc_data_list
- vc_thread_flag
- send_vc_data_thread
- vc_play_flag
- audio_data_lock
- play_vc_data_thread
- vc_data_fragments_list
- share_screen_data_fragments_list
- share_camera_data_fragments_list
- listen_udp
- listen_udp_thread
- is_watching_screen
- watching_user
- watching_type
- is_screen_shared
- send_share_screen_thread
- is_camera_shared
- send_camera_data_thread
- regular_profile_image_path
- blueish_background_color
- blackish_background_color
- reddish_background_color
- grayish_background_color
- special_design_color

- blueish_style_hover_color -
- blackish_style_hover_color -
- reddish_style_hover_color -
- grayish_style_hover_color -
- special_design_hover_color -
- hex_hover_colors -
- hex_colors -
- color_design_options -
- color_design_mapping -
- style_color_hover_mapping -
- special_keys_mapping -
- reversed_keys_mapping -
- standard_hover_color -
- background_color_hex -
- font_options -
- blur_effect -
- screen_width -
- screen_height -
- is_create_group_pressed -
- is_create_group_inside_chat_pressed -
- is_rename_group_pressed -
- is_add_users_to_group_pressed -
- current_search_song_dict -
- phone_number -
- email -
- messages_font_size -
- volume -

- output_device_name -
- input_device_name -
- camera_index -
- font_size -
- font_text -
- background_color -
- sensor_data_from_strangers -
- is_private_account -
- push_to_talk_key -
- two_factor_authentication -
- playlist_volume -
- playlist_songs -
- playlist_index -
- playlist_last_index -
- shuffle -
- replay_song -
- size_error_label -
- is_chat_box_full -
- is_friends_box_full -
- is_last_message_on_screen -
- list_messages -
- is_user_have_current_chat_all_messages -
- request_list -
- is_in_a_call -
- is_calling -
- calling_to -
- in_call_with -

- is_getting_called -
- getting_called_by -
- is_joining_call -
- joining_to -
- call_dicts -
- username -
- selected_chat -
- is_current_chat_a_group -
- camera_devices_names -
- mute -
- deafen -
- selected_settings -
- is_push_to_talk -
- is_push_to_talk_pressed -
- is_editing_push_to_talk_button -
- profile_pic -
- list_user_profile_dicts -
- circular_images_dicts_list_of_users -
- circular_images_dicts_list_of_groups -
- is_watching_video -
- messages_content_saver -
- is_messages_need_update -
- online_users_list -
- friends_list -
- friends_box_page -
- chats_list -
- file_to_send -

- file_name -
- chat_start_index -
- Network -
- chat_box_chats_index -
- chat_box_index_y_start -
- friends_box_index -
- friends_box_index_y_start -
- current_friends_box_search -
- selected_group_members -
- create_group_index -
- add_users_to_group_index -
- group_max_members -
- blocked_list -
- groups_list -
- chat_clicked -
- social_clicked -
- setting_clicked -
- music_clicked -
- is_new_chat_clicked -
- current_chat_box_search -
- temp_search_list -
- spacer -
- updated_chat_signal -
- updated_requests_signal -
- updated_settings_signal -
- getting_call_signal -
- stop_sound_signal -

- initiating_call_signal -
- reset_call_var_signal -
- new_message_play_audio_signal -
- stop_watching_stream_signal -
- caching_circular_images_of_users_signal -
- caching_circular_images_of_groups_signal -
- disconnect_signal -
- updating_profile_dict_signal -
- update_group_lists_by_group -
- update_message_box_signal -
- scroll_back_to_index_before_update_signal -
- insert_messages_into_message_box_signal -
- insert_new_message_in_chat_signal -
- insert_search_result_signal -
- close_call_threads_signal -
- start_call_threads_signal -
- insert_playlist_to_table_signal -
- update_settings_from_dict_signal -
- update_chat_page_without_messages_signal -
- sound_effect_media_player -
- mp3_message_media_player -
- calling_media_player -
- playlist_media_player -
- ringtone_media_player -
- ringtone -
- ding_sound_effect -
- new_message_audio -

- chat_box
- main_layout
- friends_box
- stacked_widget
- music_box

תפקיד ושימוש: חלק ממחלקות הGUI, המחלקה אחראית על מסך האפליקציה הראשי. מסך בו באמת הכל מתנהל. דפים מתחלפים אל תוך הדף הזה ביניהם לפי המשתמש. המחלקה אחראית על כל הדפים שנטענים אל תוך דף זה. כלומר היא מחזיקה בהמון מידע שחשוב עבור דפים אלו. המחלקה אחראית על כל שינוי בGUI לאחר ההתחברות כל הודעה מהשרת שמשפיעה על המסך עוברת דרך מחלקה זו והפונקציות שלה.

שם הפונקציה	טענת כניסה	טענת יציאה
close_all_threads	אין	משנה את כל הערכים של הדגלים של threads הקיימים לFalse. ובכך סוגרת אותם.
thread_play_vc_data	אין	thread האחראי על השמע של השיחה. מקבל מידע של אודיו ומשמיע אותו למשתמש במידת הצורך.
thread_send_voice_chat_data	אין	thread שאחראי על שליחת האודיו שיוצא מהמיקרופון בעת שיחה
thread_send_share_screen_data	אין	thread שאחראי על שליחת הצילום מסך בעת סטריים של שיתוף מסך של הלקוח

thread_send_share_camera_data	אין	thread שאחראי על שליחת הפריימים של המצלמה בעת סטריים של שיתוף מצלמה של הלקוח
listen_udp_socket_thread	אין	thread שמקשיב לכל ההודעות שמתקבלות מsocket udp ומעביר אותם הלאה לפונקציות המתאימות.
handle_udp_data	מידע	מטפל בפקטות udp
handle_vc_data	מידע	מקבל מידע מסוג אודיו, אם יש צורך בהרכבה מרכיב אם לא מעביר הלאה לרשימת של מידע האודיו.
handle_stream_data	מידע	מרכיב את המידע, לאחר הרכבה מעביר את המידע אל שמירת הנתונים של הפריימים.
handle_data_fragment	רשימת מידע נתון, מידע חדש, האם הגיע ראשון, האם אחרון, סוג המידע	מקבל סוג של מידע קוד. יודע לאן הוא שייך וממין אותו בהתאם אחראי על ההרכבה המלאה של הפקטות שנשלחות כסלייסים.
exit_group	תעודת זהות של הקבוצה	שולחת הודעה לשרת כדי לצאת מהקבוצה הנתונה
remove_friend	שם החבר	שולחת הודעה לשרת בשביל להסיר את החבר
send_friend_request_for_user	שם החבר	שולחת הודעה לשרת כדי לבקש חברות עם החבר שהפונקציה קיבלה.
remove_user_from_group	תעודת הזהות של הקבוצה, שם הלקוח להסיר	הפונקציה שולחת הודעה לשרת כדי להסיר את החבר מהקבוצה הנתונה.

right_click_object_func	מיקום, הורה, כפתור, רשימת פעולות, שם הצ'אט (לא חובה), תעודת הזהות של קבוצה (לא חובה)	הפונקציה אחראית לייצר אפשרות ללחיצת ימינית על אובייקט. היא תייצר את הלחיצה ותחבר אותה לפונקציה נתונה לפי שם הפעולה.
update_slider_position	מיקום חדש	מעדכן את המיקום של הגרר על הסליידר של המוזיקה.
toggle_shuffle	אין	משנה את הערך של shuffle ל-True אם False והפוך.
remove_song_from_playlist	אין	מוריד את השיר הנבחר באותו הרגע מהפלייליסט ושולח לשרת לעדכן את זה גם.
music_button_clicked	אין	מעביר לחלון של אזור המוזיקה.
insert_search_result	מילון החיפוש	מכניס אל תוך טבלת החיפוש את התוצאות שקיבל מהשרת.
play_search_result	אין	אם קיימות תוצאות חיפוש משמיע את האודיו של התוצאה.
save_searched_song_to_playlist	אין	מוסיף את השיר שכרגע חופש על הפלייליסט.
is_song_exist_in_playlist	המילון של השיר	בודק אם שיר בעל אותו שם קיים בפלייליסט אם כן מחזיר True אם לא False.
insert_playlist_to_table	אין	מכניס את כל השירים שקיבל מהשרת אל תוך טבלת השירים בדף המוזיקה.
pause_and_unpause_playlist	אין	אם כרגע נגן הפלייליסט משמיע עוצר אותו והפוך.

אם קיים שיר בפלייליסט עם index שקטן מ1 מהindex של השיר כרגע משמיע אותו.	אין	go_to_last_song
אם קיים שיר בפלייליסט עם index שגדול מ1 מהindex של השיר כרגע משמיע אותו.	אין	go_to_next_song
מקבל את שינויי הסטטוס של נגן הפלייליסט במידה ומגיע לסוף של שיר, משמיע את השיר הבא בפלייליסט.	סטטוס	handle_playlist_song_state_change
מעביר לשיר במקום האינדקס בפלייליסט ומשמיע אותו.	אינדקס	set_new_playlist_index_and_listen
מבקש מהשרת את הביטים של שיר בפלייליסט לפי אינדקס.	אין	play_playlist_by_index
מחזיר מילון של כל ההגדרות עם הערכים שלהם.	אין	get_setting_dict
בעזרת המילון מעדכן את המשתנים שקשורים להגדרות ומעדכן את הGUI	מילון של הגדרות	update_settings_from_dict
שולח לשרת את מילון ההגדרות במעודכן על מנת לעדכן.	אין	update_settings_dict
מתחיל את הthread שמקשיב לsocket הudp.	אין	start_listen_udp_thread
מתחיל את הthreads שנפתחים בתחילת שיחה קולית.	אין	start_call_threads
סוגר את הthreads שנפתחים בתחילת שיחה קולית.	אין	close_call_threads

סוגר את threadn שמשמיע את מידע האודיו שמתקבל מהשרת.	אין	close_listen_thread
סוגר את threadn שאחראי לשלוח את מידע האודיו אל השרת.	אין	close_send_vc_thread
מתחיל את threadn שמשמיע את מידע האודיו שמתקבל מהשרת.	אין	start_listen_thread
מתחילת את threadn שאחראי לשלוח את מידע האודיו אל השרת.	אין	start_send_vc_thread
לאחר הוספה של ווידג'טים מחזיר את המיקום של מסך ההודעות למיקום המקורי לפני ההוספה.	מספר הווידג'טים האחרונים	scroll_back_to_index_before_update
לוקח את רשימת ההודעות ומוסיף אותם ל-GUI הצ'אט	רשימת של מילוני הודעות	insert_messages_into_message_box
מוסיף הודעה בתחילת הצ'אט	מילון של הודעה	insert_new_message_in_chat
מוסיף לצ'אט הודעה שהלקוח עצמו שלח	מילון של הודעה	insert_message_that_client_send
גולל עד למטה בדף ההודעות של הצ'אט הנבחר	אין	scroll_maximum_down
מעדכן את ה-GUI של דף המוזיקה.	אין	update_message_box
עובר על רשימת מילוני הקבוצות עד שמוצא את המילון שהוא צריך להחליף ואז מעדכן אותו בכך שמחליף + עדכון GUI.	מילון של קבוצה	update_groups_list_by_dict

מעדכן את ערך הווליום כל הנגנים הקשורים לערך שהתקבל (יש הרבה נגנים שונים לא כולם על אותו ערך ווליום)	ערך חדש של ווליום	update_media_players_volume
עוצר את נגן האודיו של הקבצים אם הוא פועל וממשיך אותו עם הוא עוצר.	אין	pause_or_unpause_mp3_files_player
מעדכן את כל הGUI	אין	update_every_screen
משנה את הצבע הנבחר של העיצוב ומעדכן את הGUI	הצבע החדש	update_background_color
אם קיימת ברשימה מחזיר את התמונה העגולה של המשתמש	שם המשתמש	get_circular_image_bytes_by_name
אם התמונה של הקבוצה קיימת מחזירה אותה.	תעודת זהות הקבוצה	get_circular_image_bytes_by_group_id
לוקח את התמונות של הלקוחות, מעגל אותם בשביל עיצוב ושומר אותם בצורה העגולה.	אין	caching_circular_images_of_users
לוקח את התמונות של הקבוצות, מעגל אותם בשביל עיצוב ושומר אותם בצורה העגולה.	אין	caching_circular_images_of_groups
אם המילון כבר קיים מחליף בין המילון הישן למועדכן או לא קיים מוסיף אותו.	שם הלקוח לעדכון, מילון הפרופיל החדש	update_profile_dict_of_user
מעדכן את התמונה של לקוח ושומר אותה	שם, הביטים של התמונה החדשה, אם קיימים הביטים של התמונה העגולה	update_profile_pic_dicts_list

מעדכן את תמונה עגולה של לקוח ושומר אותה	שם, הביטים של התמונה החדשה, אם קיימים הביטים של התמונה העגולה	update_circular_photo_of_user
מעדכן את תמונה עגולה של קבוצה ושומר אותה	תעודת הזהות של הקבוצה, הביטים של התמונה החדשה, אם קיימים הביטים של התמונה העגולה	update_circular_photo_of_group
מחזיר את מילון הפרופיל המתאים לשם המשתמש שהפונקציה קיבלה.	שם משתמש	get_profile_pic_by_username
מחליף לעמוד הנבחר הנכון לפי מה שהמשתמש לחץ	אין	set_page_index_by_clicked
יוצא מהדף שבה המשתמש צופה בסרטונים	אין	stop_watching_video
נכנס אל הדף שבו מנגונים סרטונים, ומנגן את הסרטון לפי הביטים שלו.	הביטים של הסרטון	start_watching_video
מתחיל את הthread האחראי לשלוח את הפריימים של צילומי המסך.	אין	start_share_screen_send_thread
מתחיל את הthread האחראי לשלוח את הפריימים של המצלמה.	אין	start_camera_data_thread
מעדכן את הthread האחראי לשלוח את הפריימים של צילומי המסך.	אין	update_share_screen_thread
מעדכן את הthread האחראי לשלוח את הפריימים של המצלמה.	אין	update_share_camera_thread
עוצר ומסיים את הthread האחראי לשלוח את הפריימים של המצלמה.	אין	end_share_camera_thread

מקבל פריים של הסטריים שכרגע הלקוח צופה בו, ומעדכן אותו על הGUI.	פריים	update_stream_screen_frame
יוצא מהמסך בו צופים בסטריים של לקוח	איו	stop_watching_video_stream
נכנס מהמסך בו צופים בסטריים של לקוח	אין	start_watching_video_stream
מחזיר את המנהל של הקבוצה הקיימת עם התעודת זהות הזו.	תעודת הזהות של הקבוצה	get_group_manager_by_group_id
מחזיר את השם של הקבוצה הקיימת עם התעודת זהות הזו.	תעודת הזהות של הקבוצה	get_group_name_by_id
מחזיר False אם לא קיימת שיחה עם התעודת זהות של הקבוצה הזו וTrue אם קיימת.	תעודת הזהות של קבוצה	is_call_dict_exist_by_group_id
מחזיר את מספר המשתתפים של קבוצה עם התעודת זהות שתקבלה	תעודת הזהות של קבוצה	get_number_of_members_by_group_id
מחזיר את המשתתפים של קבוצה עם התעודת זהות שתקבלה	תעודת הזהות של קבוצה	get_group_members_by_group_id
מסיר את המילון של השיחה עם התעודת זהות הזו	תעודת הזהות של השיחה	remove_call_dict_by_id
מחזיר מילון של שיחה עם תעודת הזהות של הקבוצה שהפונקציה קיבלה.	תעודת הזהות של קבוצה	get_call_dict_by_group_id
מחזיר את מילון השיחה בו נמצא השם משתמש בין משתתפי השיחה.	שם משתמש	get_call_dict_by_user

is_call_dict_exists_by_id	תעודת הזהות של השיחה	אם מילון של שיחה עם תעודת הזהות הזו קיים מחזיר True אם לא מחזיר False
update_call_dict_by_id	המילון של שיחה מעודכנת	מעדכן את המילון הישן של אותה שיחה
reset_call_var	אין	מחזיר את כל המשתנים שקשורים לשיחה לערכם הראשוני.
end_current_call	אין	יוצא מהשיחה שכרגע פועלת ומודיע לשרת.
initiate_call	אין	מתקשר לצ'אט שכרגע נחבר, שולח הודעה מתאימה לשרת
handle_state_changed_sound_effect	מצב	אם המצב של הנגן שאחראי על ניגון הרינגטון משתנה משמיע מחדש את הצלצול.
play_ding_sound_effect	אין	משמיע אפקט אודיו של דינג (פעמון)
getting_a_call	אין	משמיע אל תוך נגן הרינגטון את הרינגטון
new_message_play_audio	אין	משמיע את האפקט הקולי של קבלת הודעה חדשה
play_sound_effect	הסאונד	מקבל סאונד ומשמיע אותו על תוך נגן האפקטים.
play_calling_sound_effect	הסאונד	מקבל סאונד ומשמיע אותו על תוך נגן ההתקשרות.
stop_sound	אין	עוצר את כל הגגנים הבסיסיים

עובר למסך הצ'אט	אין	chat_clicked
עובר למסך ההגדרות	אין	settings_clicked
עובר למסך החברים	אין	social_clicked
מעדכן את מסך החברים	אין	updated_social_page
מעדכן את מסך ההגדרות	אין	updated_settings_page
פונקציה בנויה בqt5 שמקבלת על כל שחרור כפתור אירוע. הפונקציה אחראית על מצב הpush to talk כדי לראות מתי הלקוח משחרר כפתור זה.	אירוע	keyPressEvent
פונקציה בנויה בqt5 שמקבלת על כל לחיצת כפתור אירוע. מקבל את לחיצות המלקדת של הלקוח ופועלת בהתאם על פי ההגיון בקוד.	אירוע	keyPressEvent
פונקציה בנויה בqt5 שמקבלת על כל הזזה של הגלגלת אירוע. מתאפיינת בהזזת הצ'אטים הנתונים וגלילה בחלק מן המסכים הנתונים.	אירוע	wheelEvent
מחביא את מסך הצ'אט	אין	hide_chat
מראה את מסך הצ'אט	אין	show_chat
מעדכן את מסך הצ'אט כולו	אין	updated_chat
מעדכן את מסך הצ'אט ללא תיבת ההודעות.	אין	update_chat_page_without_messages
מעדכן את כל דף הצ'אט או רק את חלקו לפי הפרמטר.	מקבל האם רוצים לעדכן את כל הדף או רק חלק	update_chat_page

מעדכן את הvalues של דף MainPage כאשר לקוח מתחבר.	אין	update_values
עוצר את כל הנגנים כולל הכל.	אין	stop_all_media_player
הפונקציה נקראת כאשר הלקוח סוגר את החלון. במצב זה האפליקציה כולה נסגרת לחלוטין.	אירוע	closeEvent

שם המחלקה: ChatBox

תכונות המחלקה:

- screen_width
- screen_height
- Network
- parent
- is_getting_called
- square_label
- width_of_chat_box
- height_of_chat_box
- file_dialog
- file_name
- image_height
- image_width
- chats_buttons_list
- border_labels
- buttons_style_sheet
- call_button_style_sheet
- draw_message_start_x
- friends_button_height
- create_group_open

- create_group_open_x -
- create_group_open_y -
- text_entry -
- square_label -
- around_name -
- around_name_delta -
- call_profiles_list -
- current_chat -
- current_group_id -
- ringing_square_label -
- send_image_button -
- send_image_y -
- send_image_x -
- stop_calling_button -
- incoming_call_label -
- caller_label -
- pop_up_label -
- accept_button -
- reject_button -
- share_camera_on_icon -
- share_camera_off_icon -
- share_camera_button -
- deafened_icon -
- not_deafened_icon -
- end_call_button -
- add_user_button -
- rename_group -
- edit_group_image_button -
- join_call_button -
- message_labels -
- filename_label -

- garbage_button
- image_too_big
- border_label
- find_contact_text_entry
- friends_button

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. דף זה הוא למעשה תת דף של הדף הראשי MainPage הדף כולל בתוכו את האזור של הצ'אט - אפשרות לשיחה באמצעות הודעות - שיחות קוליות, אפשרות לstreaming, יצירת קבוצות ושליטה על התקשורת בין משתמשים. המחלקה מאופיינת בעיצוב גבוה ומביאה מין שער לתת דפים האחרים. נחשבת לתת דף המרכזי כי רוב הפעולות נעשות בתת דף זה.

שם הפונקציה	טענת כניסה	טענת יציאה
add_user_to_group_pressed	אין	הפונקציה מראה ומחביאה את האזור בו המשתמש יכול ליצור קבוצה משלו, ולמלא אותה בחברים שלו.
change_group_image	אין	הפונקציה פותחת את הfile explorer רק עבור תמונות כדי שהמשתמש יכול להחליף תמונה לקבוצה הנבחרת.
create_custom_in_call_button	רוחב, אורך, מיקום, פונקציה	הפונקציה יוצרת כפתור במיקום מסויים וגודל על פי מה שקיבלה. ובנוסף מחברת אליו פונקציה ומחזירה את הכפתור.
put_call_icons_on_the_screen	אין	כאשר הלקוח בשיחה הפונקציה שמה את כל התמונות של האנשים בשיחה על המסך.

create_watch_stream_button	רוחב, אורך, מיקום, סוג סטריים	הפונקציה מכינה כפתור במידע ויש לקוח שעושה סטריים. על ידי לחיצת הכפתור הלקוח יכול לצפות בסטריימים אחרים.
watch_stream_button_pressed	שם הלקוח, סוג הסטריים	הלחיצה שולחת הודעה לשרת כי הלקוח מנסה לצפות בstream של הלקוח הנתון. ומעבירה אותו לדף המתאים לצפייה.
create_profile_button	מיקום, שם, מילון הפרופיל	מייצר תווית של פרופיל יחד עם תמונה של לקוח
create_top_page_button	מיקום, path לאייקון של הכפתור	הפונקציה מייצרת כפתורים שאחראים לפונקציות השונות באפשר לבצע על בצי'אט מסויים.
stop_calling	אין	כאשר הלקוח מתקשר הפונקציה עוצרת את הצלצול ומודיעה כך לסרבר.
create_group_clicked	אין	הפונקציה מראה ומחביאה את האזור בו המשתמש יכול ליצור קבוצה משלו, ולמלא אותה בחברים שלו.
toggle_checkbox	ווידג'ט של pyqt5	מתקבל כאשר ווידג'ט של toggle נלחץ ומשנה את הסימון שלו. אם יש V מוריד אותה אם איו מוסיף.
handle_create_group_index	סוג הפעולה	במידה ויש ללקוח הרבה חברים הוא יכול להחליף את הדף כדי לראות כל פעם כמות מסויימת של חברים כאשר הוא מכין קבוצה חדשה.

יוצר קבוצה מהמשתתפים שנבחרו על המסך.	אין	create_dm_pressed
כאשר נלחץ כפתור ההוספה החברים שנבחרו יוספו לקבוצה שכרגע הלקוח עליה.	אין	add_users_to_group
כאשר מקבל מצב חיובי הוא מוריד את החבר שקשור לכפתור מרשימת הנבחרים. אם המצב שלילי הוא מוסיף אותו.	מצב	friend_checkbox_changed
בודק האם העכבר נמצא על רשימת הצא'טים.	מיקום העכבר	is_mouse_on_chats_list
הפונקציה נקראת כל פעם שהטקסט בחיפוש משתנה כשהוא משתנה כך גם filter של החיפוש והGUI מתעדכן.	אין	on_text_changed_in_contact_search
מחזיר רשימה מופולטרת מרשימה הצ'אטים.	אין	return_search_list
מעלה את הווידג'טים הנבחרים הכי למעלה בדף כדי שיהיה אפשר לראות אותם.	אין	raise_needed_elements
מכין את כפתור chatn שם את השם של החבר התמונה שיש לו. במידה וזו קבוצה ולא משתמש יהיה את מספר המשתתפים בה.	טקסט, מיקום	create_friend_button
הפונקציה מכינה את כפתור המשתתפים בקבוצה, הכפתור דומה	טקסט, מיקום	create_member_button

לכפתור הצ'אט אך יש לו פעולות אחרות.		
מרים את התווית ליד שם הצ'אט הנבחר.	אין	raise_around_name_label
פותח את file explorer על מנת שינוי תמונת הקבוצה.	אין	open_file_dialog_for_changing_group_image
פותח את file explorer בשביל שהמשתמש יוכל לבחור file מסויימת לשלוח בצ'אט.	אין	open_file_dialog
אם הגודל של file תקין עובר הלאה לשליחת הקובץ.	file של path	file_to_bytes
מנסה להיכנס לשיחה של הקבוצה שהיא כרגע גם הצ'רט הנבחר. מבקש מהשרת כניסה.	אין	join_call
יוצא מהשיחה שהוא כרגע נמצא בה.	אין	end_current_call
אם הלקוח מושתק אז מבטל הפוך הלקוח הופך להיות מושתק. שולח הודעה לשרת.	אין	mute_and_unmute
אם הלקוח מוחרש אז מבטל הפוך הלקוח הופך להיות מוחרש. שולח הודעה לשרת.	אין	deafen_and_undeafen
שולח הודעה לשרת על התחלה של stream מסוג מצלמה	אין	share_camera_and_unshare
שולח הודעה לשרת על התחלה של stream מסוג שיתוף מסך	אין	share_screen_and_unshare

accept_call	אין	מאשר את השיחה תוך שליחת הודעה לשרת. מחכה לאישור מהשרת להתחלת השיחה.
reject_call	אין	מאפס את משתני השיחה. ומנתק את השיחה - שולח הודעה מתאימה לשרת.
ringing_user	שם משתמש	משנה את משתני התחלת הצלול
call_user	אין	מתחיל להתקשר לצ'אט הנבחר ושולח הודעה לשרת.
on_friend_button_clicked	שם משתמש נלחץ	מלחיצ צ'אט במידע והמשתמש בנלחץ הוא לא הצ'אט כרגע.
load_image_from_bytes_to_label	תמונה כביטים, תווית	טוען תמונה אל תוך תווית.
load_image_from_bytes_to_button	תמונה כביטים, כפתור	טוען תמונה אל תוך כפתור.
show_context_menu	מיקום, כפתור, ביטים של קובץ, סוג, שם	מחבר פונקציה של כפתור ימיני אל תוך הכפתור שהתקבל. אל ידי קליק ימיני תיהיה אפשרות להוריד את הקובץ המחובר.
create_temp_message_label	הודעה	יוצר תווית שבה יד את ההודעה שהתקבלה כפרמטר.
check_editing_status	אין	בודק האם הלקוח כותב הודעה. מחזיר True אם כן False אם לא.
garbage_button_clicked	אין	מוחק את הקובץ שהרגע נטען אל תוך שליחת ההודעות בצ'אט.

selected_chat_changed	שם הצ'אט החדש	מחליץ לצ'אט החדש, מודיע לשרת.
is_mouse_on_chat_box	מיקום עכבר	האם העכבר נמצא על תיבת הצ'אט. אם כן מחזיר True אחרת False.

שם המחלקה: MessageBox

תכונות המחלקה:

- parent
- width
- height
- main_page_object
- x
- y
- scroll_area
- layout
- space_between_widgets

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. דף זה הוא חלק מהעיצוב של דף הצ'אט, המחלקה מכינה widget שבתוכו ייכנסו כל ההודעות מהצ'אט הנבחר. הפונקציה אחראית על על להראות את ההודעות ומביאה נוחות למשתמש לקריאת הצ'אט שלו באופן יעיל וברור.

שם הפונקציה	טענת כניסה	טענת יציאה
-------------	------------	------------

init_ui	אין	הפונקציה טוענת על הדף את האזור אליו ייכנסו ההודעות ומארגנת את הlayout של הווידג'ט עצמו.
scroll_maximum	אין	הפונקציה גוללת את הצ'אט הכי למטה שאפשר
update_scroll_area_parent	הורה חדש	מעדכן את הווידג'ט ההורה של הווידג'טים הנמצאים במחלקה
load_all_message_func	רשימת של מילוני הודעות	טוען את כל הרשימה לתוך אזור ההודעות
add_message_to_layout	מילון של הודעה	מוסיף אתה ההודעה הכי למטה באזור ההודעות
insert_message_to_layout	מילון של הודעה	מוסיף אתה ההודעה הכי למעלה באזור ההודעות
insert_messages_list_to_layout	רשימת של מילוני הודעות	מוסיף אתה ההודעות הכי למעלה באזור ההודעות
add_or_insert_message_to_layout	הודעה, האם להוסיף בסוף או בהתחלה	לפי הפרמטר הפונקציה מוסיפה את ההודעה או להתחלה או לסוף
clear_layout	אין	מנקה את הדף
update_messages_layout	אין	מעדכן את הדף
scroll_to_index	אינקס	גולל לפי האינדקס שקיבל
scroll_up_by_n_widgets	מספר טבעי	גולל למעלה לפי הגודל של ה-N ההודעות האחרונות
scroll_value_changed	ערך	משנה את הערך של הגלגלת לערך החדש

שם המחלקה: CreateGroupBox

תכונות המחלקה:

- parent
- x
- y
- box_format
- create_group_open_x
- create_group_open_y
- selected_group_members
- group_max_members
- friends_list
- standard_hover_color
- create_group_index
- page_plus_selected_label
- amount_of_people_to_add_text_label

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. דף זה הוא הדף בוא המשתמש יכול ליצור קבוצה, הדף עובד בצורה מאוד פשוטה - המשתמש רואה את החברים שלו ויכול לעבור ביניהם, תוך כדי הוא יכול לבחור את מי שהוא רוצה ואז ללחוץ על כפתור כדי ליצור קבוצה. הדף משמש גם להוסיף משתמשים חדשים לקבוצה קיימת. הדף הוא תת דף לדף ChatBox ומופיע בו.

שם הפונקציה	טענת כניסה	טענת יציאה

init_ui	אין	הפונקציה טוענת על הדף את כל הכפתורים והתוויות ההתחלתיים. ומוסיפה אותם ל-GUI.
update_labels_text	אין	במידה ונתונים השתנו הפונקציה נקראת ומעדכנת את הטקסט בתווית השונות.

שם המחלקה: FriendsChatListWidget

תכונות המחלקה:

- chat_box_object
- friends_button_height
- draw_friends_buttons

תפקיד ושימוש: מחלקה זו היא חלק מה-GUI של הלקוח. המחלקה הזו אחראית ליצור ווידג'ט של רשימת כל החברים. למעשה זו המחלקה תיקח את כל הצ'אטים הקיימים של המשתמש ותימור עבורם כפתורים. המחלקה מקלה מאוד על לגלול בין צ'אטים מכיוון שאפשר לשנות את המיקום מאוד בקלות.

שם הפונקציה	טענת כניסה	טענת יציאה
draw_friends_buttons	רשימת הצ'אטים	הפונקציה יוצרת את כל הכפתורים של הצ'אטים וטוענת אותו אל הדף.

שם המחלקה: SettingsBox

תכונות המחלקה:

- font_size -
- parent -
- Network -
- settings_button_height -
- file_dialog -
- privacy_button_width -
- privacy_button_height -
- default_labels_font_size -
- my_account_button -
- user_profile_button -
- appearance_button -
- voice_video_button -
- privacy_safety_button -
- log_out_button -
- combo_box_style_sheet -
- volume_slider_style_sheet -
- profile_image_label -
- volume_slider -
- volume_label -
- output_combobox -
- input_combobox -
- camara_devices_combobox -
- color_combobox -
- font_size_label -

- font_size_slider
- font_box
- profile_image_label

תפקיד ושימוש: מחלקה זו היא חלק מה-GUI של הלקוח. המחלקה מחזיקה בה את כל העיצוב והמימוש הטכני של דף ההגדרות. בדף ההגדרות הלקוח יכול לשנות דברים על פי רצונו. הדף מתחלק לכמה תתי דפים שמוחזקים גם הם במחלקה זו.

שם הפונקציה	טענת כניסה	טענת יציאה
input_device_changed	אין	מעדכן את input device שהשתנה
output_device_changed	אין	מעדכן את output device שהשתנה
camera_device_changed	אין	מעדכן את האינדקס של המכשיר מצלמה
create_privacy_labels	מיקום התחלתי, רשימה של הטקסטים, מרחק בין תוויות	יוצר כמות מסויימת של תוויות על פי הפרמטרים במרחק מסויים זה מזה.
create_privacy_buttons	מיקום התחלתי, רשימה של רשימה של משתנים, רשימה של משתנים של parent, מרחק בין תוויות	יוצר כפתורים עבור התת עמוד של הפרטיות, מקבל את המשתנים שהוא רוצה לקרוא להם וגם את המשתנים אליהם הכפתורים ייקשרו.
switch_off_variable_plus_change_icon	כפתור, שם משתנה	הכפתורים מקושרים למשתנים כאשר המשתנים שונים, כך גם נראות הכפתור שמקושר למשתנה הזה.
background_color_changed	אין	מעדכן את הצבע החדש שנבחר

font_updated	אין	מעדכן את font החדש שנבחר
remove_profile_pic	אין	מסיר את תמונת הפרופיל של המשתמש
edit_profile_pic_pressed	אין	כאשר המשתמש רוצה לשנות תמונה פרופיל נפתח file explorer למטרה זו.
open_file_dialog	אין	פותח את file explorer
font_size_changed	הגודל החדש	משנה את הגודל של ההודעות בתיבת ההודעות.
create_my_account_labels	מיקום, גודל font, טקסט 1, טקסט 2	יוצר תוויות עוקבות אחת אחר השנייה למטרת עיצוב התת דף לשינוי הפרטים.
change_input_mode	אין	מחליף בין האם push to talk להאם לא.
create_colored_button	צבע של הרקע, צבע העבירה על הכפתור, צבע במסגרת, מיקום, רוחב, גובה, טקסט	יוצר כפתור לפי הפרמטרים שהפונקציה מקבלת
create_select_push_to_talk_key	מיקום	יוצר את כפתור בן ניתן לראות את כפתור האם push to talk כרגע. ניתן גם לשנות אותו על ידי כפתור זה.
handle_push_to_talk_selection_button_clicked	אין	בודק האם המשתמש מנסה לבחור כפתור חדש ל push to talk
push_to_talk_label	מיקום, גודל font, אורך, רוחב, טקסט, צבע, צבע מסגרת.	יוצר תווית עובר האם push to talk לפי הפרמטרים

יוצר את הכפתורים לבחירת דיבור רגיל או push to talk.	מיקום	create_input_mode_select_button
יוצר קופסאת בחירה על פי הפרמטרים. יהיה אפשרות לבחור בכל האיברים ברשימה. מחזיר את הקופסא הזו.	מיקום, גודל, רשימה של איברים	create_option_box
מחזיר תווית עם התמונה הנתונה.	path לתמונה, גודל, מיקום.	create_image_label
יוצר תווית עם טקסט בצבע לבן לפי הפרמטרים ומחזיר אותה.	גודל, מיקום, גודל font, טקסט	create_white_label
יוצר את התווית לפי הפרמטרים ומחזיר אותה.	מיקום, גודל font, גודל, טקסט, צבע, בולד או לא	create_custom_label
מעבירה לאזור לשינוי שם המשתמש	אין	change_username_function
מעבירה לאזור לשינוי הסיסמא	אין	change_password_function
מעבירה לאזור של מחיקת המשתמש	אין	delete_account_function
מעדכן את ערך הווליום עם הערך החדש.	ערך	set_volume
עובר לעמוד המשתמש שלי	אין	my_account_pressed
עובר לעמוד הפרופיל שלי	אין	user_profile_pressed
עובר לעמוד העיצוב.	אין	appearance_pressed
עובר לעמוד האודיו וידאו.	אין	voice_video_pressed
עובר לעמוד הפרטיות	אין	privacy_safety
יוצר את הכפתורים הראשיים של המחלקה. מחזירה אותם.	טקסט, פונקציה לחיבור, מיקום.	create_settings_main_buttons

generate_button_stylesheet	צבע רגיל, צבע עבירה, צבע לחיצה.	יוצר משתנה עיצוב לפי הפרמטרים ומחזיר אותו.
----------------------------	---------------------------------	--

שם המחלקה: CustomComboBox

תכונות המחלקה:

המחלקה יורשת ממחלקה קיימת בqt5 בשם QComboBox.

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. בqt5 יש אפשרות לבנות ווידג'ט שאתה לוחץ עליו אתה רוצה בחירות מסוימות כמו מין - תיבת בחירות כזו. איך בתיבה רגילה כזו אין אופציה לקבל signal כאשר התיבה נלחצת אז יצרתי את זה בעצמי כהוספה לclass הקיימת.

פונקציות שהוספתי למחלקה הקיימת:

שם הפונקציה	טענת כניסה	טענת יציאה
showPopup	אין	מחזיר signal כאשר הקופסה נפתחת.
hidePopup	אין	מחזיר signal כאשר הקופסה נסגרת.

שם המחלקה: VideoPlayer

תכונות המחלקה:

- parent
- video_bytes
- media_player
- video_widget
- slider

- duration_label
- media_player
- position_timer

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. מחלקה זו אחראית על צפייה בסרטונים הקיימים בצ'אט ההודעות. המחלקה מקבלת ביטים של סרטון - פותחת דף משלה לצפייה בסרטון. אפשר לחזור לדפים הקודמים או להמשיך לצפות לפי רצון הלקוח.

שם הפונקציה	טענת כניסה	טענת יציאה
update_slider_position	אין	מחזיר signal כאשר הקופסה נפתחת.
toggle_play_pause	אין	מחזיר signal כאשר הקופסה נסגרת.
stop_watching	אין	סוגר את החלון ומפסיק לצפות
play_video	אין	מתחיל את הסרטון
update_duration	משך	מעדכן את הזמן שעבר
update_position	מיקום	מעדכן את המיקום של הגררה של הסליידר
handle_state_change	מצב	מנהל האם הסרטון נגמר להתחיל אותו מחדש.
set_position	אין	מעדכן את המיקום של הסליידר לפי הsignal של timer
keyPressEvent	אירוע	אם הייתה לחיצת כפתור בודק האם יש לצאת מהסרטון חזרה לדף הראשי.

שם המחלקה: PlaylistWidget

תכונות המחלקה:

- parent -
- sliders_style_sheet -
- last_selected_row -
- search_table -
- table -
- search_song_entry -
- add_to_playlist_button -
- try_searched_song_button -
- remove_selected_song_button -
- playlist_duration_slider_duration_label -
- playlist_duration_slider_current_time_label -
- playlist_duration_slider -
- shuffle_button -
- replay_song_button -
- playlist_volume_slider_label -
- playlist_volume_slider -

תפקיד ושימוש: מחלקה זו היא חלק מהGUI של הלקוח. מחלקה זו היא אחראית על דף המוזיקה עם הפלייליסט. בדף זה הלקוח יכול לבחור שירים לשמוע או להוסיף שירים חדשים לפי בחירתו. יכול לשמוע להעביר לשירים אחרים.

שם הפונקציה	טענת כניסה	טענת יציאה

init_ui	אין	טוען את כל הווידג'טים הראשוניים אל המסך במיקומים הרצויים.
update_media_player_position	מיקום חדש	מעדכן את מיקום הסליידר
update_current_duration_text	טקסט	מעדכן את הטקסט של תווית הזמן
update_duration_tex	טקסט	מעדכן את הטקסט של תווית הזמן הכולל
set_icon_for_volume_label	אין	מעדכן את האייקון של תווית הווליום
playlist_volume_update	ערך	מעדכן את ערך הווליום של גן הפלייליסט
toggle_shuffle	אין	מעדכן את ערך shuffle אם הוא True ל False ולהפך.
toggle_replay_song	אין	מעדכן את ערך replaysong אם הוא True ל False ולהפך.
remove_song	אין	מוחק את השיר הנבחר מהפלייליסט.
create_table_widget	גודל, מיקום, שם.	יוצר טבלה לפי הפרמטרים ומחזיר אותה.
update_music_page_style_sheet	אין	מעדכן את העיצוב לכל הווידג'טים
apply_style_sheet_to_text_entry	אין	מעדכן את העיצוב לתוויות
apply_style_sheet_for_button	אין	מעדכן את העיצוב לכפתורים
apply_table_stylesheet	אין	מעדכן את העיצוב לטבלאות

מחזיר את השורה הראשונה שמתאימה לטקסט - אם אף אחד לא מתאימה מחזיר -1.	טקסט	find_row_by_text
מכניס אל תוך טבלת החיפוש שיר שחופש.	מילון של שיר	insert_search_data
מכניס את כל השירים אל תוך טבלאת הפלייליסט	רשימה של מילוני השירים בפלייליסט	insert_playlist_songs
מכניס את השיר אל סוף הטבלה של השירים.	מילון של שיר	insert_new_song_to_playlist
כאשר תא נלחץ הפונקציה נקראת ומטפלת בזה.	שורה, עמודה	cell_pressed
בוחר שורה ומדגיש אותה.	שורה	select_row
מוריד את ההדגשה על כל השורות.	אין	clear_selection
כאשר המשתמש בוחר משהו חדש - מקבל אירוע ומטפל בו.	אין	onSelectionChanged

שם המחלקה: ClientNet

תכונות המחלקה:

- logger
- client_tcp_socket
- client_udp_socket
- server_ip
- port

- addr
- original_len
- size
- mtu
- aes_key
- connected
- sending_tcp_data_lock

תפקיד ושימוש: המחלקה מהווה חלק מרכזי בפרוטוקול התקשורת של הלקוח. המחלקה אחראית על ניהול הסקוטים והחיבור עם השרת. מנסחת הודעות מקצה לקצה - מצפינה אותם ועושה את החלפת המפתחות עם השרת.

שם הפונקציה	טענת כניסה	טענת יציאה
if_connected	אין	מנסה להתחבר אל השרת
connect_tcp	אין	מנסה להתחבר על סוקט tcp
connect_udp	אין	מנסה להתחבר על סוקט udp
send_bytes	מידע	שולח את המידע כbytes אל השרת כtcp
send_large_udp_data	מידע, סוג מידע, צורת הפריים	שולח מידע של קudp במידע והוא גדול מחלק אותו לכמה הודעות קטנות.
send_bytes_udp	מידע	שולח את המידע כbytes אל השרת בudp
check_max_packet_size_udp	אין	בודק את mtu של רשת הלקוח.

שולח את המילון כקט	מילון של הודעה	send_message_dict_udp
שולח את המילון כקט	מילון של הודעה	send_message_dict_tcp
שולח לשרת הודעה מנוסחת שבעזרתה השרת יודע שהסוקט קט והסוקט tcp שייכים לאותו משתמש.	אין	connect_between_udp_port_address_to_username
שולח חיפוש של שיר אל השרת	טקסט חיפוש	send_song_search
שולח בקשה להסיר שיר על פי שמו מהפלייליסט	שם השיר	send_remove_song_from_playlist
שולח בקשה לשמור את השיר בפלייליסט	מילון השיר	save_song_in_playlist
מבקש מילון של שיר על פי האינדקס שלו בפלייליסט	אינדקס	ask_for_song_bytes_by_playlist_index
מעדכן אצל השרת באיזה צ'אט הלקוח נמצא	הצ'אט כרגע	updated_current_chat
מבקש עוד הודעות מהצ'אט בו הלקוח נמצא כרגע	אין	ask_for_more_messages
שולח על התחלת שיתוף מסך	אין	start_screen_stream
שולח על סיום שיתוף מסך	אין	close_screen_stream
שולח על התחלת שיתוף מצלמה	אין	start_camera_stream
שולח על הפסקת שיתוף מצלמה	אין	close_camera_stream
שולח הודעה על מנת להתחיל לצפות בסטריים של הלקוח מסוג שיתוף מסך	שם לקוח	watch_screen_stream_of_user

שולח הודעה על מנת להתחיל לצפות בסטריים של הלקוח מסוג מצלמה	שם לקוח	watch_camera_stream_of_user
מודיע לשרת כי מפסיק לצפות בשיתוף שהוא כרגע צופה בו	אין	stop_watching_current_stream
מודיע לשרת כי הוא רוצה לעזוב את השיחה	אין	leave_call
מעדכן את הסיסמא הישנה לסיסמא חדשה אצל השרת	סיסמא חדשה	send_new_password
מעדכן תמונה חדשה אצל קבוצה מסויימת עבור השרת.	התמונה כביטים, תעודת הזהות של הקבוצה	send_new_group_image_to_server
שולח בקשה ליצור קבוצה עם רשימת משתתפים	רשימת המשתתפים	create_group
מבקש להוסיף רשימה של אנשים לקבוצה עם הת"ז הנתון.	רשימת לקוחות, תעודת הזהות של הקבוצה	add_user_to_group
שולח הודעה להודיע כי הוא מתקשר ללקוח	הלקוח שאליו מתקשרים	send_calling_user
מפסיק לצלצל למי שהוא כרגע מצלצל אליו מעדכן את השרת.	אין	stop_ringing_to_group_or_user
מבקש להצטרף לשיחה עם קבוצת בעלת הת"ז הנתון	תעודת הזהות של הקבוצה	send_join_call_of_group_id
הודעת אישור - עונה לשיחה הנתונה	מי שמתקשר	send_accept_call_with
הודעת ביטול - לא לענות לשיחה הנתונה.	מי שמתקשר	send_reject_call_with

משנה את ערך mute אצל השרת עבור המשתמש	אין	toggle_mute_for_myself
משנה את ערך deafen אצל השרת עבור המשתמש	אין	toggle_deafen_for_myself
שולח בקשה להתחבר	שם, סיסמא	send_login_info
שולח בקשה להירשם	שם, סיסמא, אימייל	send_sign_up_info
מנסה להתחבר על ידי תוקן האבטחה השמור על המחשב	תוקן אבטחה	send_security_token
שולח מידע בסיסי ללא הסיסמא במטרת להחליף סיסמא	שם משתמש, סיסמא	send_username_and_email_from_get_password
שולח הודעה חדשה אל הרשת בצ'אט.	שולח, נשלח אל, תוכן ההודעה, סוג, שם הקובץ	send_message
שולח את התמונה החדשה אל השרת	התמונה כביטים	send_profile_pic
שולח את האודיו אל הרשת.	מידע האודיו	send_vc_data
שולח אותו אל השרת.	מידע השיתוף מסך, צורת הפריים	send_share_screen_data
שולח אותו על השרת.	מידע המצלמה, צורת הפריים	send_share_camera_data
מעדכן את מילון ההגדרות אצל השרת	מילון ההגדרות	send_settings_dict_to_server
שולח את בקשת החברות אל השרת	שם הלקוח שהוא מציע לו חברות	send_friend_request
מוריד את המשתמש מהחברים שלו	שם המשתמש	send_remove_friend

מבקש לצאת מהקבוצה עם הת"ז הנתון.	תעודת הזהות של הקבוצה	send_exit_group
מבקש להסיר את הצ'אט מרשימת הצ'אטים	שם הצ'אט	send_remove_chat
מבקש להוציא את המשתמש מהקבוצה עם הת"ז הנתון.	שם הלקוח, תעודת הזהות של הקבוצה	send_remove_user_from_group
מבקש לחסום אותו	שם הלקוח	block_user
מבקש לבטל חסימה עבור הלקוח.	שם הלקוח	unblock_user
מבקש לסרב לבקשת החברות שלו	שם הלקוח	send_friends_request_rejection
מבקש לאשר את בקשת החברות	שם הלקוח	send_friends_request_acceptio n
מבקש לקבל את תוקן האבטחה בכדי לשמור את המשתמש על המכשיר.	אין	ask_for_security_token
שולח את קוד האישור של תהליך ההירשמות	קוד	send_sign_up_verification_code
שולח את קוד האישור של תהליך ההתחברות בעזרת 2fa	קוד	send_login_2fa_code
שולח הודעת יציאה	אין	send_logout_message
מקבל הודעה על פי הגודל המבוקש	גודל	receive_by_size
מקבל הודעה שצפויה להיות מחורזת מקודדת - מפענח במידת הצורך	אין	recv_str
מקבל הוא שצפויה להיות bytes - מחזיר אותה אחרי פענוח	אין	recv_bytes

recv_udp	אין	מקבל הודעה שצפויה להיות bytes מסוקט הקט udp - מחזיר אותה
return_socket	אין	מחזיר את סוקט הקט tcp
close	אין	סוגר את כל הסוקטים
initiate_rsa_protocol	אין	מתחיל את פרוטוקול החלפת המפתחות עם השרת.

שם המחלקה: ServerNet

תכונות המחלקה:

- logger
- client_tcp_socket_address
- server
- size
- original_len
- aes_key
- sending_tcp_data_lock

תפקיד ושימוש: המחלקה מהווה חלק מרכזי בפרוטוקול התקשורת של השרת. המחלקה אחראית על ניהול הסוקטים והחיבור עם הלקוח. מנסחת הודעות מקצה לקצה - מצפינה אותם ועושה את החלפת המפתחות עם הלקוח.

שם הפונקציה	טענת כניסה	טענת יציאה
-------------	------------	------------

get_aes_key	אין	מחזיר את המפתח להצפנת aes עם הלקוח.
receive_by_size	אין	מקבל הודעה על פי הגודל המבוקש
send_str	מידע	שולח מידע שהוא מחרוזת כמחרוזת מקודדת.
send_bytes	מידע	שולח את המידע כביטים ומצפין אותו אם קיים מפתח.
send_message_dict_tcp	מילון של הודעה	שולח את המילון אל הלקוח
send_played_song_bytes	הביטים של השיר, שם השיר	שולח את השיר אל הלקוח.
send_searched_song_info	המילון של השיר המחופש	שולח את תוצאת חיפוש השיר אל הלקוח
send_settings_dict	מילון ההגדרות	שולח אתה המילון אל הלקוח
send_messages_list	רשימת מילונים של הודעות צ'אט	שולח אותם אל הלקוח
send_addition_messages_list	רשימת מילונים של הודעות צ'אט	שולח אותם אל הלקוח
send_new_message_content	מילון של הודעה, צ'אט	שולח אותם אל הלקוח
send_requests_list	רשימה של בקשות	שולח אותם אל הלקוח
sent_code_to_mail	אין	מודיע כי הקוד נשלח לאימייל
sent_friend_request_status	סטטוס	בעת בקשת חברות מחזיר את הסטטוס הנתון
send_vc_data	מידע האודיו, הדובר	שולח אותם אל הלקוח

שולח אותם אל הלקוח	מידע השיתוף מסך, השולח, צורת הפריים	send_share_screen_data
שולח אותם אל הלקוח	מידע המצלמה, השולח, צורת הפריים	send_share_camera_data
מודיע ללקוח כי הוא יכול לצאת ממסך הטעינה היא כל המידע נשלח.	אין	send_to_client_he_has_all_of_the_messages
שולח צ'אט להוסיף ללקוח	צ'אט להוסיף	add_new_chat
שולח אותו ללקוח להוסיף	מילון הקבוצה	send_new_group
שולח אותו ללקוח לעדכן	מילון הקבוצה	update_group
שולח אותם אל הלקוח	רשימת החברים	send_friends_list
שולח אותם אל הלקוח	רשימת של מילוני שירי הפלייליסט	playlist_songs_list
שולח אותם אל הלקוח	רשימת החסומים	send_blocked_list
שולח אותם אל הלקוח	רשימת החברים המחוברים	send_online_users_list
שולח אותם אל הלקוח	רשימת מילוני הקבוצות	send_user_groups_list
שולח אותם אל הלקוח	רשימת הצ'אטים	send_user_chats_list
שולח הודעה מתאימה	המשתמש במתקשר אל הלקוח	send_user_that_calling
מודיע שהשיחה שהלקוח נמצא בה הסתיימה	אין	send_user_that_call_ended
מודיע שהשיחה שכרגע שהלקוח מחכה אליה התחילה.	אין	send_user_that_call_accepted

מודיע כי הצלצול של השיחה הסתיים.	אין	send_user_call_timeout
שולח את המילון אל הלקוח ומעדכן אותו שם.	מילון של שיחה	send_call_dict
שולח אל הלקוח את כל השיחות הקיימות.	רשימה של מילוני השיחה	send_call_list_of_dicts
שולח ללקוח את הפרופילים הרלוונטיים.	רשימה של מילוני הפרופיל	send_profile_list_of_dicts
מעדכן או מוסיף אצל הלקוח מילון חלש של פרופיל.	מילון הפרופיל, משתמש	send_profile_dict_of_user
מודיע ללקוח להסיר את השיחה כי כבר לא קיימת.	תעודת הזהות של השיחה	remove_call_to_user_of_id
מודיע למשתמש כי לקוח סגר את הסטריים שלו.	שם הלקוח	send_stream_of_user_closed
מודיע כי פרטי ההתחברות נכונים	אין	send_confirm_login
מודיע כי פרטי ההתחברות אינם נכונים	אין	send_invalid_login
אם פרטי ההתחברות נכונים אך מישהו אחר מחובר למשתמש מודיע כי זהו המצב.	אין	send_already_logged_in
מודיע כי ה2fa של הלקוח פועל.	אין	send_2fa_on
מודיע כי ההרשמות עברה בהצלחה	אין	send_sign_up_confirm
מודיע כי ההרשמות נכשלה.	אין	send_sign_up_invalid
מודיע כי קוד האישור להרשמות לא נכון	אין	send_sign_up_code_invalid

מודיע כי קוד האישור להרשמות נכון	אין	send_sign_up_code_valid
מודיע כי כל המידע הגיע ואפשר להמשיך הלאה	אין	send_all_data_received
שולח אותו אל הלקוח בעת ביקושו	טוקן האבטחה	send_security_token_to_client
מודיע כי תוקן האבטחה נכון	אין	send_security_token_valid
מודיע כי תוקן האבטחה לא נכון	אין	send_security_token_invalid
מודיע כי המידע של השכחתי סיסמא הוא נכון	אין	send_forget_password_info_val id
מודיע כי המידע של השכחתי סיסמא הוא לא נכון	אין	send_forget_password_info_inv alid
מודיע כי השכחתי סיסמא קוד נכון	אין	send_forget_password_code_v alid
מודיע כי ה2fa קוד נכון	אין	send_2fa_code_valid
מודיע כי ה2fa קוד לא נכון	אין	send_2fa_code_invalid
מודיע כי השכחתי סיסמא קוד לא נכון	אין	send_forget_password_code_in valid
שולח הודעה לtimeout threads שנתקעים על .recv	אין	timeout_receive
מקבל הודעה שמצופה להיות מחרוזת מקודדת. - מחזיר את המחרוזת	אין	recv_str
מצפה לקבל הודעה מורכבת מbytes - מחזיר את ההודעה אחרי פענוח	אין	recv_bytes

return_socket	אין	מחזיר את tcp socket
close	אין	סוגר את הסקוט
initiate_rsa_protocol	אין	מתחיל את פרוטוקול החלפת המפתחות עם הלקוח.

מסמך בדיקות מלא:

שם הבדיקה	מטרת הבדיקה	מה בוצע בפועל	מה היו תוצאות הבדיקה	בעיות שהתגלו וכיצד נפתרו
התחברות לקוח	למנוע כניסה לאתר עם נתונים שגויים	יצרתי hash עם הסיסמא יחד עם salt ו-pepper בצד השרת והשוואתי אותו עם hash שהיה שמור במסד הנתונים	הבדיקה איפשרה לי לדעת האם הנתונים שהוכנסו היו שגויים או לא וכך יכלתי לפעול בהתאם (להכניס את המשתמש או להוציא) (הודעת שגיאה מתאימה)	לא היו
בדיקת תקינות האימייל	למנוע ניסיון הרשמות עם אימייל לא תקין	בדקתי האם קיימים דפוסים מסוימים שקיימים בכל המיילים	בעזרת הבדיקה עצרתי את ניסיון ההרשמות עם אימייל לא נכון או נתתי להירשם עם אימייל אפשרי	לא היו
בדיקת תקינות הסיסמא	למנוע ניסיון הרשמות עם סיסמא קצרה מדי	בדקתי את אורך הסיסמא	אם הסיסמא קצרה מדי לא נתתי אפשרות להירשם אחרת ניתנה אפשרות	לא היו
בדיקה האם	למנוע כניסה	נשלחת הודעת בקשה	אם כאשר ההודעה	לא היו

	אפשר להיכנס לשיחה	לשיחה שהסתיימה	לשרת להיכנס לשיחה	מגיעה לשרת השיחה עדיין פועלת נשלחת הודעת אישור ללקוח אם לא נשלחת הודעת האישור.
בדיקת הוספת שיר לפלייליסט	למנוע כפילויות של שירים	בדקתי האם שם השיר זהה לשם שקיים כבר בפלייליסט	אם יש שם זהה לא ניתן להוסיף את השיר, אם אין השיר יוסף	לא היו
בדיקת הצעת חברות	למנוע כפילויות של הצעות חברות	בודק אם ההצעת חברות כבר קיימת או עם המשתמשים כבר חברות או אם מישהו חסום	במידה והם חברות או חסומים מתקבלת הודעה מתאימה ומופיעה הודעת שגיאה. במידע ולא ההצעת חברות עובדת כרגיל.	לא היו
בדיקה האם לקוח כבר מחובר	למנוע כניסה של 2 לקוחות אל אותו המשתמש	בודק האם הפרטי זיהוי מופיעים במשתמשים המחוברים	במידה והמשתמש כבר מחובר לא ניתן גישה כפולה ומראה הודעת שגיאה מתאימה. במידה והוא לא מחובר מתחבר כרגיל.	לא היו
בדיקת תקינות שם המשתמש בעת יצירת לקוח	למנוע של 2 לקוחות יהיה אותו שם משתמש	בודק אם השם משתמש כבר קיים במסד הנתונים	אם קיים לא ניתן ליצור, מופיעה הודעת שגיאה. במידה והשם משתמש לא קיים ממשיך את תהליך ההרשמה.	לא היו
בדיקת קבלת מידע מסוג אודיו	למנוע שליחה של אודיו למקום הלא נכון.	בודק אם הלקוח ששלח את מידע האודיו בהכרח נמצא	אם הלקוח לא נמצא בשיחה מידע האודיו יאבד ותישלח הודעה	לא היו

	ללקוח. במידע והוא בשיחה המידע יעבור הלאה למשתמשים האחרים.	בשיחה.		
--	--	--------	--	--

מדריך למשתמש

פירוט כלל קבצי המערכת - עץ קבצים:

דף עצים ללקוח:

```
.
├── project_files/
│   ├── chat_file.py
│   ├── client_net.py
│   ├── Discord_app_client.py
│   ├── messages_page_widgets.py
│   ├── settings_page_widgets.py
│   ├── social_page_widgets.py
│   ├── song_search_engine.py
│   └── discord_app_assets/
│       ├── accept_button.png
│       ├── add_image_button.png
│       ├── add_user.png
│       ├── block_icon.png
│       ├── camera_icon.jpg
│       ├── camera_icon.png
│       ├── camera_watch_icon.png
│       ├── connectify_icon.png
│       ├── deafened.png
│       ├── deafened_profile.png
│       ├── Ding Sound Effect.mp3
│       ├── Discord mute sound effect.mp3
│       ├── down_arrow_icon.png
│       ├── edit_image_icon.png
│       ├── edit_name.png
│       ├── exit_button.png
│       ├── friends_icon.png
│       ├── garbage_icon.png
│       ├── Getting_called_sound_effect.mp3
│       ├── hide_password_icon.png
│       ├── hide_password_icon1.png
│       ├── info_icon.png
│       ├── join_call_sound_effect.mp3
│       ├── last_song_icon.png
│       ├── leave_call_sound_effect.mp3
│       ├── mic_muted_icon.png
│       ├── mic_not_muted_icon.png
│       ├── monitor_icon.png
│       ├── music_icon.png
│       ├── muted_profile.png
│       ├── new_message_sound_effect.mp3
│       ├── next_song_icon.png
│       ├── not_deafened.png
│       ├── not_select_circle.png
│       ├── no_camera_icon.png
│       ├── off_button.png
│       ├── on_button.png
│       ├── pause_and_play_icon.png
│       ├── pause_icon.png
│       ├── Phone Internal RingingCalling - Sound Effect.mp3
│       ├── play_video_icon.png
│       ├── press_chat_icon.png
│       ├── regular_profile.png
│       ├── reject_button.png
│       ├── remove_friend_icon.png
│       ├── replaying_icon.png
│       ├── right-arrow-icon-27.png
│       ├── ringing_blue_icon.png
│       ├── select_circle.png
│       ├── Setting_logo.png
│       ├── share_screen_off_icon.png
│       ├── share_screen_on_icon.png
│       ├── show_password_icon.png
│       ├── speaker_icon.png
│       ├── speaker_icon0.png
│       ├── speaker_icon1.png
│       ├── speaker_icon2.png
│       ├── speaker_icon3.png
│       ├── shuffle_icon.png
│       ├── up_arrow_icon.png
│       └── white_pause_icon.png
```

עץ קבצים לשרת:

```
.
└─ project_files/
    ├── connectify_db.sqlite
    ├── database_func.py
    ├── discord_server.py
    ├── email_send_code.py
    ├── server_handling_classes.py
    ├── server_net.py
    └─ discord_app_files/
```

התקנת המערכת:

כדי להריץ את הפרויקט יש צורך בהורדת interpreter של python 3.8.0, בנוסף לכך יש צורך יש צורך בשני מחשבים או יותר הנמצאים על אותה רשת. אפשר להריץ גם על אותו מחשב.

הקבצים הנחוצים להרצת הפרויקט:

לכל צד (השרת) הצגתי עץ קבצים, כל קבצים אלו נחוצים להרצת הפרויקט בצורה תקינה.

סביבת עבודה:

את הפרויקט ניתן להריץ מכל קומפיילר של פייתון כל עוד הגרסה של ה-interpreter היא python 3.8.0, אבל ניתן גם להריץ את הפרויקט ישירות מהCMD של המחשב דרך ה-interpreter המתאים.

מיקומי קבצים:

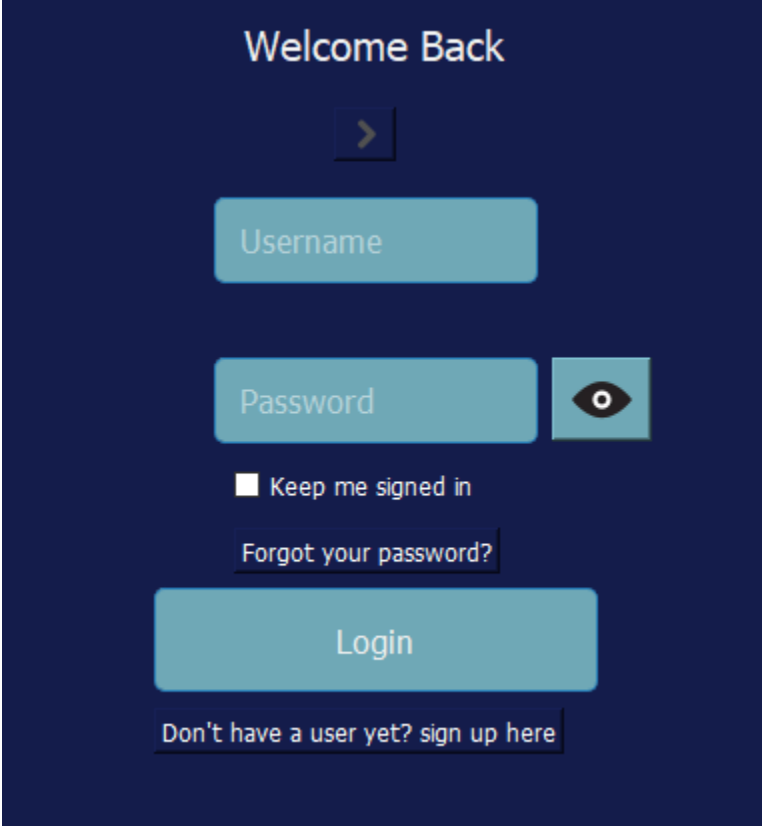
הקבצים צריכים להיות ממוקמים על פי עץ הקבצים.

אופן ההפעלה:

כדי להפעיל את המערכת יש להעלות את הקבצים של השרת והלקוח לשני מחשבים שונים על אותו רשת. לאחר שהתקנו את הקבצים על פי עץ הקבצים אפשר לעבור לשלב הבא. נלך לקובץ client_net ונשנה את כתובת הip של השרת שנמצאת במשתנה server_ip בתוך מחלקה בשם ClientNet אל הכתובת הנכונה. כיצד? במידע ומרצים אתה הלקוח והשרת על שני מחשבים שונים נצטרך לבדוק את הipv4 של השרת. נפתח CMD במחשב בו אנחנו מריצים את השרת. נכתוב את הפקודה IPCONFIG ונסתכל על התוצאות שלנו. ניקח את הכתובת של

ה IPv4 address איפה שיש ערך ל Default gateway. נשים את הכתובת בתוך הקוד ונמשיך הלאה.

כעת על מנת שהלקוח באמת יעבוד יש להריץ קודם כל את קובץ השרת discord_server לאחר מכן יש להריץ את קובץ הלקוח discord_app_client.
אם הכל עבד כמו שצריך אתם אמורים לראות את דף ההתחברות.
שם משתמש וסיסמא קיימים במסד הנתונים: kevin123, kevin123



Welcome Back

>

Username

Password

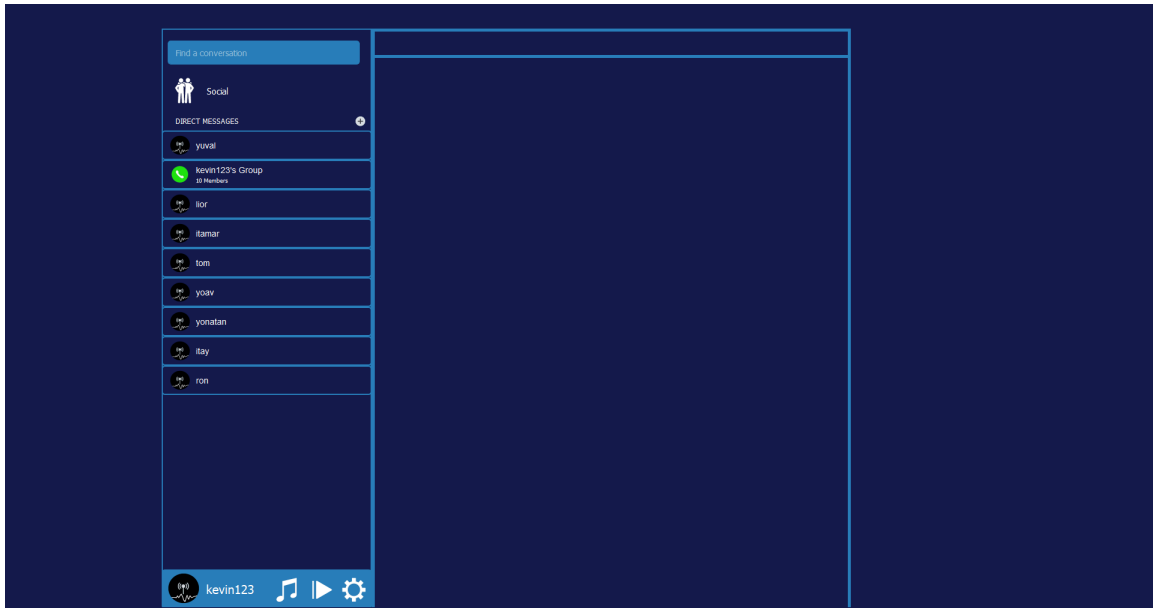
☐ Keep me signed in

[Forgot your password?](#)

Login

[Don't have a user yet? sign up here](#)

זהו מרכז הדף שאתם אמורים להיות. תכניסו את הפרטים השמורים במסד הנתונים (לא חובה אפשר גם להירשם) ולחצו על כפתור ה Login.



זהו הדף שתראו מולכם. הכפתורים העיקריים שאתה צריכים לדעת עליהם הם הכפתורים הבאים:



כפתור זה הוא כפתור המוזיקה - בלחיצה עליו הם תעברו לדף המוזיקה בו תוכלו לראות את הפלייליסט האישי שלכם ולהוסיף אליו שירים לפי בחירתכם.



כפתור ההגדרות: הכפתור יקח אותכם אל דף ההגדרות שם תוכלו לשנות את ההגדרות להגדרות הרצויות לכם.



כפתור החברים: על ידי לחיצה על כפתור זה תועברו לדף בו תוכלו לראות את החברים שלכם, אנשים שחשמתם, מי מחובר וגם להוסיף חברים חדשים.

– על ידי לחיצה על כפתור הESC אפשר לחזור לדף הראשי שהוא דף הצ'אטים.

בעת לחיצה של צ'אט אתם תוכלו לראות את היסטוריית ההודעות, לשלוח הודעה חדשות (גם קבצים מסוגים מסוימים), להתקשר ועוד.

The screenshot shows the WhatsApp interface with the following annotations:

- Find a conversation:** An arrow points to the search bar at the top left.
- Social:** An arrow points to the 'Social' icon in the left sidebar, with the text "לעבור לאזור החברים" (Go to the friends area).
- DIRECT MESSAGES:** An arrow points to the 'DIRECT MESSAGES' section in the left sidebar.
- yuval:** An arrow points to the contact 'yuval' in the left sidebar.
- Kevin123's Group:** An arrow points to the group chat 'Kevin123's Group' in the left sidebar.
- lior, itamar, tom, yoav, yonatan, itay, ron:** Arrows point to these individual contacts in the left sidebar.
- לעבור לצ'אט עם רון:** An arrow points to the contact 'ron' in the left sidebar.
- לעבור לדף המוזיקה:** An arrow points to the music icon in the bottom left corner.
- העלאת קובץ מהמחשב של הלקוח:** An arrow points to the '+' icon in the bottom right corner of the chat area.
- האזור בו אפשר לכתוב הודעות טקסט:** An arrow points to the text input field at the bottom of the chat area.
- ליצור קבוצה יחד עם משתמש הצ'אט כרגע:** An arrow points to the group icon in the top right corner of the chat area.
- על ידי לחיצה על כפתור זה ניתן להתקשר:** An arrow points to the voice call icon in the top right corner of the chat area.
- חיפוש צ'אט מסויים מתוך רשימה הצ'אטים הקיימים:** An arrow points to the search bar at the top of the chat area.
- כפתור היצירת קבוצה חדשה:** An arrow points to the '+' icon in the top left corner of the chat area.

בתמונה למעלה אפשר לראות דף שמסביר בדיוק מה כל כפתור עושה בדף הצ'אט. חשוב

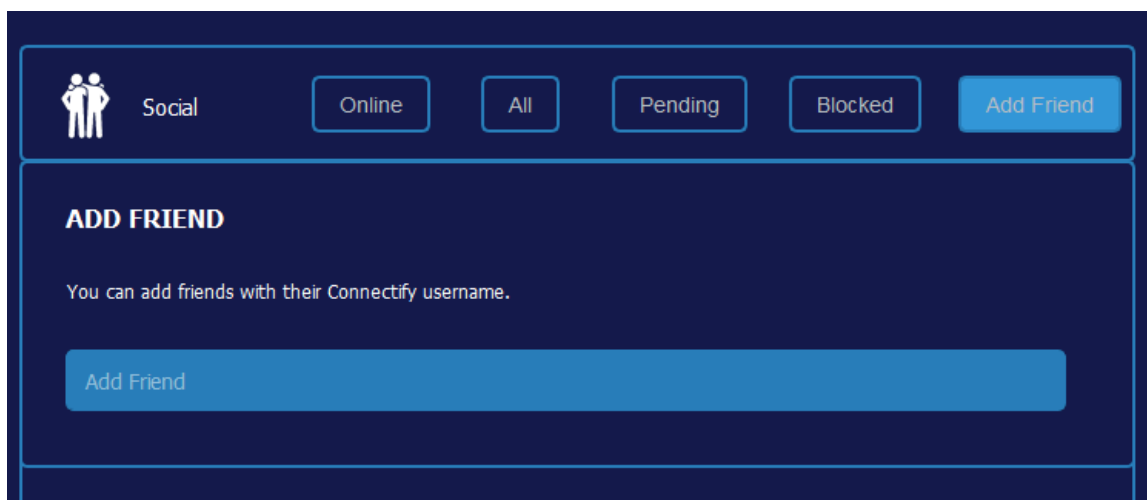
לציין כדי לשלוח הודעה עליכם ללחוץ על כפתור Enter.

בדפים האחרים אין הרבה אופציות כמו בדף הזה. הדפים האחרים מאוד ברורים וקל מאוד להבין מה קורה איתם ומה עושה מה.

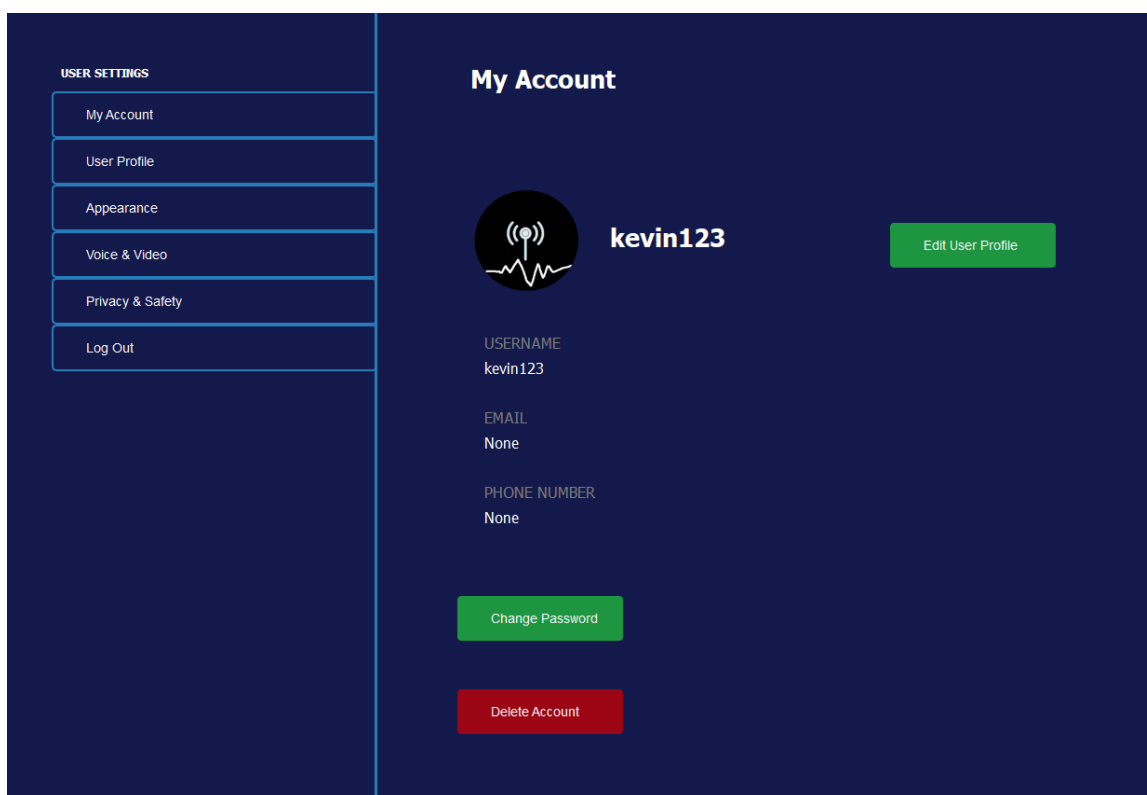


בדף החברים אתה יכול לראות את כל החברים שלך או מי מהחברים שלך שמחובר. בנוסף ניתן לראות את האנשים החסומים על ידי הלקוח ובקשות חברות שעדיין לא נמסרה עליהם תגובה.

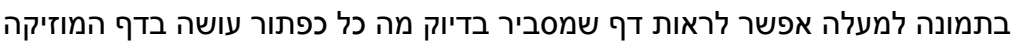
בעת ניסיון הוספת חבר על ידי לחיצה על Add Friend ככה הדף נראה.



כעת על ידי רשימת שם של משתנה ולחיצה על כפתור ה-Enter תשלח הצעת החברות.
דף ההגדרות:



כעת אתם רואים את דף ההגדרות הראשוני שהוא הפרופיל של המשתמש. יש כפתורים בצד שיביאו אותו לאופציות אחרות בהגדרות איך הם מאוד פשוטים להבנה.



רפלקציה

בתחילת השנה שהתחלתי את הפרויקט הייתי בטוח שיהיה לי מאוד קשה אך הסיבה שחשבתי שתהיה לקושי היא לא חוסר ידע אלא חוסר מוטיבציה. התחלתי לעבוד על הפרויקט בין הילדים הראשונים במגמה שלי. כבר בחודש נובמבר התחלתי להתנסות עם ממשקים גרפיים שונים ליצירת הפרויקט. באותה תקופה גם המורה שלי לסייבר אופיר רצה שנתחיל את תיקי הפרויקט - בשלב זה בכלל חשבתי שאני אעשה את האפליקציה שלי - דמוי דיסקורד כאתר אינטרנט, אך תוכניות משתנות ואין קבועות. נהנתי מלמידה מקדימה על פיתוח אתר אך לא הרגשתי שזה 100% בשבילי. במשימות שאופיר נתן לנו כהכנה לפרויקט הגמר היה המון שימוש בגרפיקה של ספריות GUI בפיתוח בעיקר Tkinter, מאוד התפעלתי מהספרייה אך לא התחברתי אליה לגמרי. המטרה שלי הייתה לייצר פרויקט שנראה עיצובית מאוד טובה ואסתטי וכשציפיתי בסרטוני הדרכה על Tkinter הבנתי שזה מאוד שאפשרי לעשות גם בספריות בפיתוח. התחלתי לערוך חיפוש ארוך מאוד אחר הספרייה שהכי מתאימה לי. חשוב לציין שלמרות הניסיון עם Tkinter במשימות שניתנו לנו בכיתה לא השתמשתי בה למרות היכולות הגבוהות היא כי לא התחברתי אל איך שהיא עובדת בכלל. לאחר המון חיפושים מצאתי ספרייה בשם `pygame`, אני זוכר את הסרטון הראשון שראית עליה ביוטיוב זה היה סרטון על GUI בעזרתה של מטבע דיגיטלי התפעלתי נורא מהעיצוב ואמרתי שאני חייב לנסות בעצמי. בהתחלה היה לי מאוד קשה למרות שהספרייה יחסית פופולרית לא היה עליה הרבה מידע באינטרנט לכן הרבה זמן הייתי צריך לנסות ולחשוב על דברים בעצמי. נהנתי מאוד מההליך עבודה איתה - היא התאימה לי בדיוק, למרות שיצירת דברים מורכבים היה קשה הספרייה הייתה מורכבת בדיוק כמו שציפיתי ובדיוק כמו שאני אוהב.

הזכרתי את ההליך הטכני אבל דלגתי קצת על ההליך האישי, חשבתי שיהיה לי מאוד קשה לעבוד על הפרויקט בגלל חוסר מוטיבציה וזמן, אבל משום מקום מצאתי צד אצלי שלא הכרתי מאוד טוב. גם לכיתה י"א קיבלנו משימה לעשות פרויקט לא קטן, פרויקט של משחק מאוד נהנתי מההליך וגם מאוד השקעתי בו אבל השנה זה היה משהו מאוד שונה הפרויקט מצא מקום מאוד קרוב ללב שלי כמעט ולא עבר יום שלא נגעתי בו מתחילת העשייה.

הלמידה לאורך הפרויקט הייתה לא קטנה, אם זה היה לצפות בסרטונים שעות על גבי שעות או חיפושים חוזרים באינטרנט אך אף פעם לא התייאשתי. היו שלבים של באסה אירוע אחד שזכור לי טוב מאוד היה כאשר היה קורה לי ניתוק בין התקשורת של הלקוח לשרת אך השרת עדיין עדכן את הלקוח בעדכונים איך לא משנה מה קרה היה נראה שהלקוח לי יכול לשלוח הודעה לשרת. עשיתי debugging שעות על גבי שעות וזה אפילו לקח כמה ימים. בהתחלה חשבתי כי יש בעיה בסוקט אך לבסוף גיליתי שהצד המקבל בשרת נכנס ללולאה אינסופית בגלל שורה אחת חסרה. אני זוכר את ההרגשה שהרגשתי היה באסה שbug קטן היה כזה קשה לתיקון אך ההרגשה העיקרית הייתה מין שמחה שהקוד סוף סוף עובד ואפשר להמשיך אל האתגר הבא.

למדתי המון על עצמי לאורך הפרויקט ואחד הדברים העיקריים הם כנראה שאני לפעמים יותר מדי נלהב, עולה לי רעיון משהו חדש להוסיף לפרויקט ובלי לחשוב פעמיים אני מתחיל לעבוד על זה. אם אני מסתכל על במבט לאחור לא יודע אם זה תוכנה חיובית או שלילית ואני אסביר. מצד אחד ההתלהבות הזו יכולה להתאפיין גם כאימפולסיביות - משמע לעשות משהו מבלי לחשוב עליו לפני. הסיבה שזה בהכרח חיסרון בפרויקט מסוג גודל כזה היא שאתה רוצה לעשות אותו באופן מסודר מבלי להוסיף דברים ללא שליטה מבלי קשר לנושא המרכזי עליו מתבסס הפרויקט. אני חושב אבל שבאופן מסויים יש גם צד חיובי שהוא העובדה שאומנם עשיתי דברים

מהר מבלי לחשוב פעמיים אבל זה דברים שעכשיו אני רואה בפרויקט ואני שמח שהוספתי אותם, אני כבר לא יכול לראות את הפרויקט אותו דבר בלעדיהם.

אם יש משהו שאני מאוד שמח שעשיתי שהקל עליי המון במשך הפרויקט הוא פרוטוקול התקשורת שלי בין הלקוח לשרת. ניסוח ההודעות היה מאוד ברור מההתחלה ככה שהיה לי מאוד קל להוסיף עוד הודעות שנשלחות ביניהם בקלות ובכך בעצם להוסיף את יכולות לאפליקציה. ברגע שהיה לי פרוטוקול ברור על איך הודעה נשלחת ואיך הודעה מתקבלת בין שני הצדדים, השאר זה פשוט למיין את ההודעות ואת הפרמטרים שלהם שיגיעו למקום הנכון בשני הצדדים.

במהלך הפרויקט הייתי צריך עזרה הרבה פעמים אם זה היה למטרות עיצוביות או למטרות טכניות בשפה פייתון, תמיד היה נחמד שהיו לי את החברים שלי להסתמך עליהם. תמיד שהייתי צריך עזרה או המלצה פניתי לחברי איתמר ליטווין - איתמר הוא בחור נבון שלרוב נתן לי את הפתרון הנכון ועזר לי רבות במיוחד לאורך הספר פרוייקט. אומנם להרבה שאלות לא היו לא תשובות אבל עדיין היה נחמד שתמיד יש מישהו שמנסה לעזור גם כשאני יודע. לבעיות עיצוביות או המלצות התייעצתי בעיקר עם אחותי התאומה - למרות שאין לה רקע בעיצוב גרפי היא בכל זאת תמיד יודעת איזה עיצוב נראה יותר טוב, היתרון נוסף שלה היא שהיא מאוד כנה ככה שהיא לא תשקר לך בפרצוף ותגיד שה GUI נראה טוב אם היא לא מאמינה בכך באמת.

אז לאחר שדיברתי על החוויות שלי לאורך החצי שנה האחרונה בוא נדבר על הנושא המרכזי והוא הפרויקט עצמו. הפרויקט עצמו הוא רעיון מאוד בסיסי, אפליקציה שהיא דמוי דיסקורד. הסיבה שבחרתי את הרעיון הזה היא כי הוא היה נראה לי מאוד מעניין, יצירת שיחות בין אנשים ושיתוף מצלמה ומסך היו נראים דברים ברמה גבוהה שאני אשמח להתעסק איתם. אני בחיי השתמשתי בדיסקורד לא מעט פעמים כדי לדבר עם חברים שלי תוך כדי ששיחקנו במשחק אחר או סתם דיברנו ואני מאוד אוהב את האפליקציה. אז לקבל הזדמנות ליצור אותה בעצמי וליישם דברים שאני מרגיש שחסר באפליקציה האמיתית הייתה הזדמנות שלא יכולתי לבזבז. דבר נוסף שאני יכול להוסיף הוא כאשר אתה יוצר פרויקט שהוא דומה למשהו שכבר קיים העבודה הרבה יותר קלה, אתה יכול לשאוף לעיצוב דומה ולראות איך הם מיישמים דברים באפליקציה שלהם. שעבדתי על הפרויקט שחשבתי על איך ליישם פונקציה חדשה ולא הייתי בטוח הייתי מחפש באינטרנט או פותח את הדיסקורד עצמו כדי לראות עם הם מיישמים ולקחת השראה מזה.

התיק פרויקט הוא חלק מאוד גדול מהפרויקט והיה לי מאוד נחמד לעבוד עליו, אך זה כנראה היה בן הדברים היותר קשים עבורי. ליישם את הדברים בקוד תוך כדי להבין בעצמי מה קורה היה לי מאוד פשוט, אך במהלך בניית התיק הבנתי שההסבר בתיק צריך להיות מאוד רציני וברור על מנת שמישהו שלא עבד על הפרויקט בעצמו יוכל להבין מה קורה בתוך הפרויקט. מאוד נהנתי לחזור על הפרויקט במהלך כתיבת התיק ולהתפעל מהדברים שעשיתי.

דברים שלמדתי עליהם המון במהלך הפרויקט היו בעיקר קשורים לרשתות. נהנתי לחקור ולקרוא על פרוטוקולי התעבורה TCP וUDP וליישם אותם בקוד שלי. UDP לדעתי הכי עניין אותי בגלל שהוא מאוד שונה מ-TCP והוא הרבה יותר אתגר אותי. על מנת להעביר נתונים גדולים הייתי צריך לעשות תהליך שנקרא fragmentation, תהליך זה נעשה ב-TCP אך הוא מיישם את זה בעצמו ומאוד נהנתי ליישם את זה בעצמי לשימוש ב-UDP. הסיבה שהעברת נתונים גדולים לא בהכרח מותרת ב-UDP היא בגלל ה-MTU כלומר ה-MTU Maximum transmission unit זה גודל מסוים שמעליו אתה לא יכול לשלוח בשליחה אחת. לכן על מנת לשלוח מידע שגדול יותר מה-MTU אתה צריך לחלק את המידע לכמות חלקים קטנים. הפנמתי את המידע שלמדתי ואני

מרגיש שהוא מאוד יועיל לי בעתיד, שלמדנו על שכבת התעבורה בכיתה מאוד לא התעניינתי אבל תוך יישום של זה עם עצמי שמתי לב שהרבה יותר כיף לי ללמוד את זה לעומק.

אני מאוד גאה בפרויקט שיצא לי ומאוד שמח שניתנה לי ההזדמנות לעשות אותו, במהלך השנתיים הראשונות שלי במגמת סייבר לא נהנתי כל כך, למדנו המון על רשתות ופרוטוקולים איך אף פעם לא עשינו מימוש מלא שלהם עם פרוייקט גמר שכזה. אני חושב שזה מאוד מה שהלהיב אותי שאחרי כל השנים האלו סוף סוף יש הזדמנות לשלב הרבה ממה שלמדנו אל פרוייקט אחד גדול. אני יכול להגיד עכשיו שאני מאוד שמח שהלכתי למגמה הזו וכי השנה הזו באמת שינתה את דעתי לחלוטין.

אני רוצה להודות לכל האנשים שתרמו לי בעשיית הפרויקט ביניהם: אמא שלי שתמיד נתנה לי חוות דעת כנה, אחותי שעזרה לי בעיצוב האפליקציה, חברים מהכיתה שתמיד היו שם בשבילי כשהייתי צריך עזרה או המלצה: איתמר ליטווין, יואב שפט, עומר חן ותום קטלר. בנוסף אני רוצה להודות לאופיר המורה שלי לסייבר שלימד אותי לאורך השנה המון דברים ותמיד עזר לי ונתן לי טיפים בעשיית הפרויקט.

ביבליוגרפיה

QT documentation QMediaPlayer

<https://doc.qt.io/qtforpython-5/PySide2/QtMultimedia/QMediaPlayer.html>

Python GUI Development Using PyQt5

<https://www.youtube.com/watch?v=MOItX2aKTGc&t=742s>

PyQt5 Tutorial - Setup and a Basic GUI Application

<https://www.youtube.com/watch?v=Vde5SH8e1OQ>

Python SQLite Tutorial: Complete Overview - Creating a Database, Table, and Running Queries

<https://www.youtube.com/watch?v=pd-0G0MigUA>

geeksforgeeks sqlite3 python documentation

<https://www.geeksforgeeks.org/python-sqlite-cursor-object/?ref=lbq>

python.org UDP socket documentation

<https://wiki.python.org/moin/UdpCommunication>

geeksforgeeks communication-protocols-in-system-design

<https://www.geeksforgeeks.org/communication-protocols-in-system-design>

RealPython sockets documentation

<https://realpython.com/python-sockets>

ספרייה

```
import sqlite3
import binascii
import os
import hashlib
import secrets
import json
from datetime import datetime
import base64
import string
import random

folder_name = "discord_app_files"
files_folder_path = folder_name
pepper = "c5b97dce"
basic_files_types = ["xlsx", "py", "docx", "pptx", "txt", "pdf", "video", "audio",
                    ["image"
                    } = default_settings_dict
volume": 50, # Default volume level"
output_device": "Default", # Default output device"
input_device": "Default", # Default input device"
camera_device_index": 0, # Default camera device"
font_size": 12, # Default font size"
font": "Arial", # Default font"
theme_color": "Blue", # Default theme color"
censor_data": False, # Default censor data setting"
private_account": False, # Default account privacy setting"
push_to_talk_bind": None, # Default push-to-talk key binding"
```

```

2fa_enabled": False # Default 2-factor authentication setting"
                                {

                                :()def current_timestamp
                                ()time_now = datetime.now
                                ('return time_now.strftime('%Y-%m-%d %H:%M:%S

                                :(def unpack_settings(variables_dict
                                ) return
                                ,["variables_dict["volume
                                ,["variables_dict["output_device
                                ,["variables_dict["input_device
                                ,["variables_dict["camera_device_index
                                ,["variables_dict["font_size
                                ,["variables_dict["font
                                ,["variables_dict["theme_color
                                ,["variables_dict["censor_data
                                ,["variables_dict["private_account
                                ,["variables_dict["push_to_talk_bind
                                ["variables_dict["two_factor_auth
                                (

                                :(def save_bytes_to_file(data_bytes, file_path
                                :with open(file_path, 'wb') as file
                                (file.write(data_bytes

```

```

:(def file_to_bytes(file_path
""".Read file bytes from the given file path"""
:try
:with open(file_path, 'rb') as file
()file_bytes = file.read
return file_bytes
:except FileNotFoundError
("{print(f"File not found: {file_path
return None
:except Exception as e
("{print(f"Error reading file: {e
return None

:(def generate_random_filename(length=24
""".Generate a random filename"""
characters = string.ascii_letters + string.digits
((random_string = ".join(random.choice(characters) for _ in range(length
return random_string

:(def save_file(file_bytes, file_path
:try
:with open(file_path, 'wb') as file
(file.write(file_bytes
("{print(f"File saved successfully at: {file_path

```



```

        :except Exception as e
        ({print(f"Error saving file: {e

:(def create_user_settings(user_id
    Connect to the database #
    ()connection = connect_to_kevindb

    Create a cursor object to execute SQL queries #
    ()cursor = connection.cursor

    Prepare the SQL query to insert a new row into settings_table #
        """ = insert_query
        INSERT INTO settings_table
        ,user_id, volume, output_device, input_device, camera_device_index)
        ,font_size, font, theme_color, censor_data, private_account
        (push_to_talk_bind, two_factor_auth
        VALUES
        (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        """

    Extract default settings values from the default_settings_dict #
    default_settings_values = [default_settings_dict[key] for key in
        [default_settings_dict

    Insert the new row with user_id and default settings #
    ((cursor.execute(insert_query, (user_id,) + tuple(default_settings_values

    Commit the transaction #

```

```
()connection.commit
```

```
Close the cursor and database connection #
```

```
()cursor.close
```

```
()connection.close
```

```
:(def remove_song(title, owner_username
```

```
:try
```

```
Connect to your MySQL database #
```

```
(owner_id = get_id_from_username(owner_username
```

```
()connection = connect_to_kevindb
```

```
()cursor = connection.cursor
```

```
"""" = select_query
```

```
SELECT mp3_file_path, thumbnail_path FROM songs
```

```
? = WHERE title = ? AND owner_id
```

```
""""
```

```
((cursor.execute(select_query, (title, owner_id
```

```
()result = cursor.fetchone
```

```
:if result
```

```
mp3_file_path, thumbnail_path = result
```

```
Construct the SQL query to delete a song from the table #
```

```
"""" = delete_query
```

```
DELETE FROM songs
```

```
? = WHERE title = ? AND owner_id
```

```
""""
```

```
Execute the SQL query with the song data #  
((cursor.execute(delete_query, (title, owner_id
```

```
Commit the transaction #  
()connection.commit
```

```
("!print("Song removed successfully
```

```
Delete the associated song files #  
        :if mp3_file_path  
            (os.remove(mp3_file_path  
            ("print("Song file deleted successfully  
        :if thumbnail_path  
            (os.remove(thumbnail_path  
            ("print("Thumbnail file deleted successfully  
        :else  
            ("print("Song not found
```

```
        :except Exception as error  
(print("Error while removing song from the table:", error
```

```
def add_song(title, mp3_file_bytes, owner_username, duration,  
              : (thumbnail_photo_bytes  
              :try
```

```
Connect to your MySQL database #  
(owner_id = get_id_from_username(owner_username  
        ()connection = connect_to_kevindb
```

```

()cursor = connection.cursor

        """ = insert_query
INSERT INTO songs (title, mp3_file_path, owner_id, duration, timestamp,
                    (thumbnail_path
                    (? ,? ,? ,? ,? ,?) VALUES
                    """

        Generate unique filenames #
        folder_path = files_folder_path
        (mp3_file_name = generate_random_filename(24
        (mp3_file_path = os.path.join(folder_path, mp3_file_name

        (thumbnail_photo_name = generate_random_filename(24
        (thumbnail_photo_path = os.path.join(folder_path, thumbnail_photo_name

        Save files #
        (save_bytes_to_file(mp3_file_bytes, mp3_file_path
        :if thumbnail_photo_bytes is not None
        (save_bytes_to_file(thumbnail_photo_bytes, thumbnail_photo_path
        :else
        thumbnail_photo_path = None

        Execute the SQL query with the song data #
        (('timestamp = str(datetime.now()).strftime('%Y-%m-%d %H:%M
        cursor.execute(insert_query, (title, mp3_file_path, owner_id, duration,
        (('timestamp, thumbnail_photo_path

        Commit the transaction #

```

```

        ()connection.commit

    (!print("Song added successfully

except Exception as error
(print("Error while adding song to the table:", error

:(def get_songs_by_owner(owner
    :try

    Connect to your MySQL database #
(owner_id = get_id_from_username(owner
    ()connection = connect_to_kevindb
    ()cursor = connection.cursor

Construct the SQL query to select songs by owner_id #
        """ = select_query
SELECT title, mp3_file_path, duration, timestamp, thumbnail_path
        FROM songs
        ? = WHERE owner_id
        """

Execute the SQL query with the owner_id parameter #
    ((,cursor.execute(select_query, (owner_id

Fetch all rows #
    ()songs_data = cursor.fetchall

```

```

        Prepare a list to store song information dictionaries #
        [] = songs

        Iterate over the fetched rows #
        :for song_data in songs_data

        Extract song data from the row #
        title, mp3_file_path, duration, timestamp, thumbnail_path = song_data
        (timestamp_datetime = datetime.fromisoformat(timestamp

        Create a dictionary to store song information #
        (thumbnail_bytes = file_to_bytes(thumbnail_path
        } = song_info
        ,title": title"
        ,audio_duration": duration"
        timestamp": timestamp_datetime.strftime("%Y-%m-%d"), # Convert"
        timestamp to string
        thumbnail_bytes": thumbnail_bytes"
        {

        Append the song information dictionary to the list #
        (songs.append(song_info

        return songs

        :except Exception as error
        (print("Error while retrieving songs by owner_id:", error
        [] return

```

```

:(def get_song_by_index_and_owner(owner, index
                                   :try
                                   Connect to your MySQL database #
                                   (owner_id = get_id_from_username(owner
                                   ()connection = connect_to_kevindb
                                   ()cursor = connection.cursor

                                   "" = select_query
SELECT title, mp3_file_path, duration, timestamp, thumbnail_path
                                   FROM songs
                                   ? = WHERE owner_id
                                   ? LIMIT 1 OFFSET
                                   ""

Execute the SQL query with the owner_id and index parameters #
((cursor.execute(select_query, (owner_id, index

                                   Fetch the row #
                                   ()song_data = cursor.fetchone

                                   :if song_data
                                   Extract song data from the row #
title, mp3_file_path, duration, timestamp, thumbnail_path = song_data
                                   (timestamp = datetime.fromisoformat(timestamp
                                   Create a dictionary to store song information #
                                   (mp3_bytes = file_to_bytes(mp3_file_path
                                   (thumbnail_bytes = file_to_bytes(thumbnail_path
                                   } = song_info

```

```

        ,title": title"
        ,audio_bytes": mp3_bytes"
        ,audio_duration": duration"
timestamp": timestamp.strftime("%Y-%m-%d"), # Convert timestamp"
                                                to string
        thumbnail_bytes": thumbnail_bytes"
                                                {

        return song_info
                                                :else

        No song found at the specified index for the owner_id #
        return None

        :except Exception as error
(print("Error while retrieving song by index and owner_id:", error
        return None

:(def get_user_settings(username
        :try

        Connect to the database #
(user_id = get_id_from_username(username
        ()db_connection = connect_to_kevindb

        Create a cursor object to execute SQL queries #
        ()cursor = db_connection.cursor
        cursor.row_factory = sqlite3.Row #
        Define the table name #
        "table_name = "settings_table

```



```

Define the SQL SELECT statement to retrieve settings for the given #
                                user_id

select_query = f"SELECT volume, output_device, input_device,
                                \ "camera_device_index, font_size
                                f", font, theme_color, censor_data, private_account,
"? = push_to_talk_bind, two_factor_auth FROM {table_name} WHERE user_id

```

```

Execute the SELECT statement with parameterized values #
                                ((,cursor.execute(select_query, (user_id

```

```

Fetch all settings rows for the given user_id #
                                [user_settings = cursor.fetchall()[0

```

```

Close the cursor and database connection #
                                ()cursor.close
                                ()db_connection.close

```

```

                                :if user_settings

Extract column names from the cursor description #
[column_names = [description[0] for description in cursor.description

```

```

Create a dictionary mapping column names to row values #
user_settings_dict = {column_names[i]: user_settings[i] for i in
                                {{{range(len(column_names

```

```

                                return user_settings_dict
                                :else

```

```
return None
```

```
:except Exception as e
```

```
    ({print(f"Error: {e
```

```
        return None
```

```
settings names: volume, output_device, input_device, camera_device_index #  
font_size, font, theme_color, censor_data, private_account, push_to_talk_key, #  
                two_factor_auth
```

```
:(def change_user_setting(user_id, setting_name, new_value
```

```
    :try
```

```
        Connect to the database #
```

```
        ()db_connection = connect_to_kevindb
```

```
        Create a cursor object to execute SQL queries #
```

```
        ()cursor = db_connection.cursor
```

```
        Define the table name #
```

```
        "table_name = "settings_table
```

```
        Define the SQL UPDATE statement to change the setting for the given #
```

```
                user_id
```

```
update_query = f"UPDATE {table_name} SET {setting_name} = ? WHERE
```

```
                "? = user_id
```

```

Execute the UPDATE statement with parameterized values #
    ((cursor.execute(update_query, (new_value, user_id

Commit the transaction #
    ()db_connection.commit

Close the cursor and database connection #
    ()cursor.close
    ()db_connection.close

except sqlite3.Error as e
    ({print(f"Error: {e
    (.print("Failed to change setting

:(def get_id_from_username(username
    Connect to the MySQL database #
    ()conn = connect_to_kevindb
    ()cursor = conn.cursor

:try
    Execute a query to retrieve the ID based on the username #
    cursor.execute("SELECT id FROM sign_up_table WHERE username = ?",
        ((username
    row = cursor.fetchone() # Fetch the first row

:if row
    If a row is found, return the ID #
    [return row[0

```

```

                                :else
                                If no row is found, return None #
                                return None
                                :except sqlite3.Error as e
Handle any errors that occur during the execution of the query #
                                (print("Error retrieving ID:", e
                                return None
                                :finally
                                Close the cursor and connection #
                                ()cursor.close
                                ()conn.close

:(def get_email_by_username(username
                                :try
Establish a connection to the database #
                                ()connection = connect_to_kevindb

Create a cursor object to execute SQL queries #
                                ()cursor = connection.cursor

Define the SQL query to retrieve the email by username #
"? = query = "SELECT email FROM sign_up_table WHERE username

Execute the query with the provided username as a parameter #
                                ((,cursor.execute(query, (username

Fetch the result (email) from the query #

```

```

        ()result = cursor.fetchone

        :if result
            email = result[0] # Extract the email from the result tuple
            return email
        :else
            return None # Username not found or email is NULL

        :except sqlite3.Error as error
            ({print(f"Error retrieving email for username '{username}': {error}
                return None

            :finally
                Close the cursor and connection #
                ()cursor.close
                ()connection.close

            :(def get_username_from_id(user_id
                :try
                    Connect to the MySQL database #
                    ()conn = connect_to_kevindb
                    ()cursor = conn.cursor

                    Execute a query to retrieve the username based on the user ID #
                    cursor.execute("SELECT username FROM sign_up_table WHERE id = ?",
                                ((, (user_id
                                row = cursor.fetchone() # Fetch the first row

```

```

                                :if row
                                If a row is found, return the username #
                                [return row[0]
                                :else
                                If no row is found, return None #
                                return None
                                :except sqlite3.Error as e
Handle any errors that occur during the execution of the query #
                                (print("Error retrieving username:", e
                                return None
                                :finally
                                Close the cursor and connection #
                                ()cursor.close
                                ()conn.close

:(def update_settings_by_dict(username, settings_dict
                                (user_id = get_id_from_username(username
                                volume, output_device_name, input_device_name, camera_index, font_size,
                                font_text, background_color, censor_data, is_private_account, push_to_talk_key,
                                )two_factor_authentication = unpack_settings
                                (settings_dict
                                (change_volume(user_id, volume
                                (change_output_device(user_id, output_device_name
                                (change_input_device(user_id, input_device_name
                                (change_camera_device_index(user_id, camera_index
                                (change_font_size(user_id, font_size
                                (change_font(user_id, font_text
                                (change_theme_color(user_id, background_color

```

```

        (change_censor_data(user_id, censor_data
        (change_private_account(user_id, is_private_account
        (change_push_to_talk_bind(user_id, push_to_talk_key
        (change_2fa_enabled(user_id, two_factor_authentication
        ("{print(f"updated whole settings for user_id {user_id

```

```

        : (def change_volume(user_id, new_volume
        (change_user_setting(user_id, "volume", new_volume

```

```

        : (def change_output_device(user_id, new_output_device
        (change_user_setting(user_id, "output_device", new_output_device

```

```

        : (def change_input_device(user_id, new_input_device
        (change_user_setting(user_id, "input_device", new_input_device

```

```

: (def change_camera_device_index(user_id, new_camera_device_index
        change_user_setting(user_id, "camera_device_index",
        (new_camera_device_index

```

```

        : (def change_font_size(user_id, new_font_size
        (change_user_setting(user_id, "font_size", new_font_size

```

```

        : (def change_font(user_id, new_font

```

```
(change_user_setting(user_id, "font", new_font
```

```
:(def change_theme_color(user_id, new_theme_color  
(change_user_setting(user_id, "theme_color", new_theme_color
```

```
:(def change_censor_data(user_id, new_censor_data  
(change_user_setting(user_id, "censor_data", new_censor_data
```

```
:(def change_private_account(user_id, new_private_account  
(change_user_setting(user_id, "private_account", new_private_account
```

```
:(def change_push_to_talk_bind(user_id, new_push_to_talk_bind  
(change_user_setting(user_id, "push_to_talk_bind", new_push_to_talk_bind
```

```
:(def change_2fa_enabled(user_id, new_2fa_value  
(change_user_setting(user_id, "two_factor_auth", new_2fa_value
```

```
:(def decode_base64(message  
(message_content = base64.b64decode(message  
    return message_content
```



```

        : (def generate_random_salt (length=8
('salt = binascii.hexlify(os.urandom(length)).decode('utf-8
        return salt

```

```

        : (def generate_token (length=16
        """

```

.Generate a secure random token

:Parameters

.(length (int): Length of the token (default is 32 characters -

:Returns

.str: The generated token -

"""

```

        (return secrets.token_hex(length // 2

```

```

        : (def hash_sha2 (string

```

Create a new SHA-256 hash object #

```

        ()sha256_hash = hashlib.sha256

```

Update the hash object with the bytes of the string #

```

        (('sha256_hash.update(string.encode('utf-8

```

Get the hexadecimal representation of the hash #

```

        ()hashed_string = sha256_hash.hexdigest

```

```

        return hashed_string

    :(def retrieve_salt_by_username(username
                                     :try
Establish a connection to the database #
        ()connection = connect_to_kevinadb

        Create a cursor #
        ()cursor = connection.cursor

        Execute the SELECT query to retrieve the salt #
        query = f"SELECT salt FROM sign_up_table WHERE username =
                                                         '{username}'"
        (cursor.execute(query

        Fetch the salt #
        ()result = cursor.fetchone

        Close the cursor and connection #
        ()cursor.close
        ()connection.close

        (Return the salt (or None if user not found #
        return result[0] if result else None

    :except sqlite3.Error as e
        ("{print(f"Error: {e
        return None

```

```

:(def retrieve_user_id_by_username(username
                                   :try

Establish a connection to the database #
    ()connection = connect_to_kevindb

Create a cursor #
    ()cursor = connection.cursor

Execute the SELECT query to retrieve the salt #
query = f"SELECT id FROM sign_up_table WHERE username =
        '{username}'"
    (cursor.execute(query

Fetch the salt #
    ()result = cursor.fetchone

Close the cursor and connection #
    ()cursor.close
    ()connection.close

(Return the salt (or None if user not found #
    return result[0] if result else None

except sqlite3.Error as e
    ("{print(f"Error: {e
    return None

```

```

        : (def login(username, password
                                : try
                                Create a connection #
                                () connection = connect_to_kevindb
                                (salt_by_user = retrieve_salt_by_username(username
                                : if salt_by_user is None
                                return False
                                (hashed_password_salt = hash_sha2(password + salt_by_user
                                hashed_password_salt_pepper = hashed_password_salt + pepper
                                Create a cursor #
                                () cursor = connection.cursor

                                Define the table name #
                                "table_name = "sign_up_table

                                Define the SQL SELECT statement to check login credentials with case #
                                sensitivity
                                """select_query = f
                                {SELECT * FROM {table_name
                                ? = WHERE username = ? AND password
                                """

                                Execute the SELECT statement with the provided username and #
                                password
                                ((cursor.execute(select_query, (username, hashed_password_salt_pepper

                                (Fetch the result (fetchone() returns None if no matching row is found #
                                () result = cursor.fetchone

```

```

        Close the cursor and connection when done #
            ()cursor.close
            ()connection.close

    Return True if the login credentials are valid, False if they are not #
        return result is not None

    :except sqlite3.Error as e
        ("{print(f"Error: {e
            return False

:(def username_exists(username
    :try
        Create a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        Define the table name #
        "table_name = "sign_up_table

    Define the SQL SELECT statement to check if the username exists #
    "? = select_query = f"SELECT * FROM {table_name} WHERE username

    Execute the SELECT statement with the username value #

```

```

        ((,cursor.execute(select_query, (username

(Fetch the result (fetchone() returns None if no matching row is found #
        ()result = cursor.fetchone

Close the cursor and connection when done #
        ()cursor.close
        ()connection.close

Return True if the username exists, False if it doesn't #
        return result is not None

        :except sqlite3.Error as e
            ({print(f"Error: {e
                return False

:(def user_exists_with_email(username, email
    :try
        Create a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        Define the table name #
        "table_name = "sign_up_table

```

Define the SQL SELECT statement to check if the username and email #
exist

```
select_query = f"SELECT * FROM {table_name} WHERE username = ?  
                "? = AND email
```

Execute the SELECT statement with the username and email values #
((cursor.execute(select_query, (username, email

(Fetch the result (fetchone() returns None if no matching row is found #
(
)result = cursor.fetchone

Close the cursor and connection when done #

(
)cursor.close
(
)connection.close

Return True if the username and email exist, False if they don't #

return result is not None
:
except Exception as e

(Handle exceptions (print, log, or raise as needed #

("{
print(f"Error: {e
return False

:(def check_security_token(token
:
:try

Establish a connection to the database #

(
)connection = connect_to_kevin
db
(
)cursor = connection.cursor

```

        Define the table name #
        "table_name = "sign_up_table

    Define the SQL SELECT statement to check if the token exists #
    select_query = f"SELECT username FROM {table_name} WHERE
        "? = security_token

    Execute the SELECT statement with the parameterized value #
    ((,cursor.execute(select_query, (token

        Fetch the result #
        ()result = cursor.fetchone

    Close the cursor and connection when done #
        ()cursor.close
        ()connection.close

        :if result
    Token exists, return the associated username #
        [return result[0
        :else
    Token doesn't exist, return False #
        return False

        :except sqlite3.Error as e
        ("{print(f"Error: {e
        return False

```



```

:(def get_security_token(username
                                :try
Establish a connection to the database #
    ()connection = connect_to_kevindb
    ()cursor = connection.cursor

    Define the table name #
    "table_name = "sign_up_table

Define the SQL SELECT statement to get the security token by username #
    select_query = f"SELECT security_token FROM {table_name} WHERE
                                "? = username

Execute the SELECT statement with the parameterized value #
    ((,cursor.execute(select_query, (username

    Fetch the result #
    ()result = cursor.fetchone

Close the cursor and connection when done #
    ()cursor.close
    ()connection.close

    :if result
Return the security token #
    [return result[0
    :else
If username not found, return None or any other suitable value #
    return None

```

```

        :except sqlite3.Error as e
            ({print(f"Error: {e
                return None

:(def insert_user(username, password, email
    :try
        ()connection = connect_to_kevinadb
        ()cursor = connection.cursor

    Generate a random salt and hash the password with the salt #
        ()salt = generate_random_salt
        password_with_salt = password + salt
    (hashed_password_with_salt = hash_sha2(password_with_salt

        Generate a security token #
        ()security_token = generate_token

        Define the table name #
        "table_name = "sign_up_table

    Define the SQL INSERT statement with parameterized queries #
    insert_query = f"INSERT INTO {table_name} (username, password, email,
        "(? ,? ,? ,? ,?) salt, security_token) VALUES

    Execute the INSERT statement with parameterized values #
    cursor.execute(insert_query, (username, hashed_password_with_salt +
        ((pepper, email, salt, security_token

```

```

        Commit the changes to the database #
            user_id = cursor.lastrowid
            ()connection.commit

    Close the cursor and connection when done #
        ()cursor.close
        ()connection.close

    (").print("User inserted successfully

    (create_user_settings(user_id

        :except sqlite3.Error as e
            ("){print(f"Error: {e
            (").print("Failed to insert user

:(def update_profile_pic(username, profile_pic_encoded
    :try
        ()connection = connect_to_kevindb
        ()cursor = connection.cursor

    :if profile_pic_encoded is not None
(profile_pic = decode_base64(profile_pic_encoded
    :else
        profile_pic = None

```

```
"table_name = "sign_up_table
```

```
update_query = f"SELECT profile_pic_path FROM {table_name} WHERE  
                "? = username
```

```
Execute the SELECT statement with the parameterized value #  
((,cursor.execute(update_query, (username
```

```
Fetch the result #  
(,result = cursor.fetchone
```

```
file_path = result[0] # Extract the file path from the tuple
```

```
:if file_path is not None  
    :try  
        (os.remove(file_path  
    :except Exception as e  
        ("print("couldn't find file path
```

```
"table_name = "sign_up_table
```

```
:if profile_pic is not None  
    folder_path = files_folder_path  
    (file_name = generate_random_filename(24  
    (profile_pic_path = os.path.join(folder_path, file_name  
    (save_bytes_to_file(profile_pic, profile_pic_path  
    update_query = f"UPDATE {table_name} SET profile_pic_path = ?  
                    "? = WHERE username  
                :else
```

```

        profile_pic_path = None

update_query = f"UPDATE {table_name} SET profile_pic_path = ?
               "? = WHERE username

Execute the INSERT statement with parameterized values #
((cursor.execute(update_query, (profile_pic_path, username

Commit the changes to the database #
        ()connection.commit

Close the cursor and connection when done #
        ()cursor.close
        ()connection.close

    :except sqlite3.Error as e
        ("{print(f"Error: {e
        ("print("Failed to insert user

:(def get_profile_pic_by_name(username
    :try

        ()connection = connect_to_kevindb
        ()cursor = connection.cursor

        "table_name = "sign_up_table

update_query = f"SELECT profile_pic_path FROM {table_name} WHERE
               "? = username

```

```

Execute the SELECT statement with the parameterized value #
    ((,cursor.execute(update_query, (username

Fetch the result #
    ()result = cursor.fetchone

Close the cursor and connection when done #
    ()cursor.close
    ()connection.close

        :if result
        [profile_pic_path = result[0
        :if profile_pic_path is None
            return None
        :else
        (profile_pic_bytes = file_to_bytes(profile_pic_path
            return profile_pic_bytes
        :else

If username not found, return None or any other suitable value #
    return None

        :except sqlite3.Error as e
        ("{print(f"Error: {e

:(def change_password(username, new_password
    :try
        ()connection = connect_to_kevinadb

```

```

        ()cursor = connection.cursor

        Generate a new salt for the user #
        ()new_salt = generate_random_salt

        Hash the new password with the new salt #
        (hashed_new_password = hash_sha2(new_password + new_salt

        Update the user's password and salt in the database #
        update_password_query = "UPDATE sign_up_table SET password = ?, salt
                                "? = = ? WHERE username
        cursor.execute(update_password_query, (hashed_new_password + pepper,
                                                ((new_salt, username

        Commit the changes to the database #
        ()connection.commit

        Close the cursor and connection when done #
        ()cursor.close
        ()connection.close

    except sqlite3.Error as e
        ("{print(f"Error: {e
        ("print("Failed to change password

    :(def is_table_exist(table_name
        ()connection = connect_to_kevindb

```

```

        ()cursor = connection.cursor
        ("){cursor.execute(f"SHOW TABLES LIKE '{table_name

Fetch the result #
        ()result = cursor.fetchone

Return True if the table exists, False if it doesn't #
        return result is not None

def add_message(sender_name, receiver_name, message_content,
                :(message_type, file_original_name
                :try

        Establish a connection to the MySQL server #
        (sender_id = get_id_from_username(sender_name
                :(")if receiver_name.startswith
                group_name, receiver_id =
        (gets_group_attributes_from_format(receiver_name
                receiver_id = int(receiver_id) * -1
                :else
        (receiver_id = get_id_from_username(receiver_name
                ()connection = connect_to_kevindb
                ()cursor = connection.cursor

SQL query to insert a message into the 'messages' table #
                :if message_type in basic_files_types
                encoded_base64_bytes = message_content
(message_content = base64.b64decode(encoded_base64_bytes
                folder_path = files_folder_path

```



```

        : (if not os.path.exists(folder_path
            (os.makedirs(folder_path
                (file_name = generate_random_filename(24
                    (file_path = os.path.join(folder_path, file_name
                        : (if os.path.exists(file_path
                            : (while os.path.exists(file_path
                                (file_name = generate_random_filename(24
                                    (file_path = os.path.join(folder_path, file_name
                                        (save_file(message_content, file_path
                                            sql_query = "INSERT INTO messages (sender_id, receiver_id,
                                                "(? , ? , ? , ? , ? , ?) message_content_path, type, file_name, timestamp) VALUES
                                                    (('timestamp = str(datetime.now()).strftime('%Y-%m-%d %H:%M
                                                        data = (sender_id, receiver_id, file_path, message_type,
                                                            (file_original_name, timestamp
                                                                : else
                                                                    sql_query = "INSERT INTO messages (sender_id, receiver_id,
                                                                        "(? , ? , ? , ? , ? , ?) message_content, type, timestamp) VALUES
                                                                            (('timestamp = str(datetime.now()).strftime('%Y-%m-%d %H:%M
                                                                                data = (sender_id, receiver_id, message_content, message_type,
                                                                                    (timestamp

```

Execute the query #

```
(cursor.execute(sql_query, data
```

Commit changes to the database #

```
()connection.commit
```

```
:except Exception as e
```

```
(print("Error in adding message:", e
```

```

                                                    :finally
Close the database connection #
                                ()cursor.close
                                ()connection.close

:(def gets_group_attributes_from_format(group_format
                                :if "(" not in group_format
                                return group_format, None
                                :else
                                ("(")parts = group_format.split
                                [id = parts[0][1
                                [name = parts[1
                                return name, id

def get_last_amount_of_messages(sender_name, receiver_name,
                                :(first_message_index, last_message_index
                                :try
                                :if first_message_index > last_message_index
                                ("print("wrong parameters
                                [] return
                                ('')is_group_chat = receiver_name.startswith

                                :if is_group_chat
(group_id = gets_group_attributes_from_format(receiver_name ,_
                                receiver_id = int(group_id) * -1
                                :else

```

```

(sender_id = get_id_from_username(sender_name
(receiver_id = get_id_from_username(receiver_name

```

```

Connect to the database using a context manager #
()connection = connect_to_kevindb

```

```

()cursor = connection.cursor

```

```

limit = last_message_index - first_message_index + 1
offset = first_message_index

```

```

:if is_group_chat

```

```

        """ = query

```

```

SELECT IFNULL(message_content, message_content_path) AS
                                                ,content

```

```

        sender_id, timestamp, type, file_name

```

```

        FROM messages

```

```

WHERE receiver_id = ? ORDER BY message_id DESC LIMIT ?
                                                ? OFFSET
                                                """

```

```

((cursor.execute(query, (receiver_id, limit, offset

```

```

        :else

```

```

        """ = query

```

```

SELECT IFNULL(message_content, message_content_path) AS
                                                ,content

```

```

        sender_id, timestamp, type, file_name

```

```

        FROM messages

```

```

WHERE (sender_id = ? AND receiver_id = ?) OR (sender_id = ? AND
        ? receiver_id = ?) ORDER BY message_id DESC LIMIT ? OFFSET
                                                """

```

```

cursor.execute(query, (sender_id, receiver_id, receiver_id, sender_id,
                      ((limit, offset

```

```

Fetch all messages within the specified range #

```

```

(messages_new_to_old = cursor.fetchall

```

```

messages = #

```

```

[messages_new_to_old[first_message_index:last_message_index + 1

```

```

(formatted_messages = format_messages(messages_new_to_old

```

```

return formatted_messages

```

```

except sqlite3.Error as err

```

```

({print(f"Error retrieving messages: {err

```

```

[] return

```

```

:(def get_username_for_senders(message_list

```

```

(sender_ids = set(message["sender_id"] for message in message_list

```

```

(sender_id_list = list(sender_ids

```

```

Fetch usernames for all unique sender_ids in one query #

```

```

{} = sender_id_to_name

```

```

:if sender_id_list

```

```

:try

```

```

(conn = connect_to_kevindb

```

```

(cursor = conn.cursor

```

```

Construct the query to fetch usernames for sender_ids #

```

```

(placeholders = ','.join('? ' for _ in sender_id_list

```

```

query = f"SELECT id, username FROM sign_up_table WHERE id IN
                                                "({{placeholders
(cursor.execute(query, sender_id_list
        ()result = cursor.fetchall

        :for row in result
[sender_id_to_name[row[0]] = row[1

        :except sqlite3.Error as err
("{print(f"Error retrieving sender usernames: {err

        :finally
        :if cursor
        ()cursor.close
        :if conn
        ()conn.close

    return sender_id_to_name

:(def format_messages(messages
(sender_id_to_name = get_username_for_senders(messages #
    [] = formatted_messages

    :for message in messages
content, sender_id, timestamp, message_type, file_name = message

    :if message_type != "string
(content_bytes = file_to_bytes(content

```

```

('content = base64.b64encode(content_bytes).decode('utf-8
    (sender_name = get_username_from_id(sender_id
        } = formatted_message
        ,content": content"
        ,sender_id": sender_name"
        ,timestamp": timestamp"
        ,message_type": message_type"
        file_name": file_name"
    {
    (formatted_messages.append(formatted_message

    return formatted_messages

```

```

:(def are_friends(username, friend_username
    Assuming you have a MySQL database connection #
    Replace 'your_database', 'your_user', 'your_password' with your actual #
    database credentials

```

```

    (username_id = get_id_from_username(username
    (friend_username_id = get_id_from_username(friend_username
    )connection = connect_to_kevindb

```

```

    ()cursor = connection.cursor

```

```

    Check if the users are already friends #
    query = f"SELECT friendship_status FROM friends WHERE (user_id =
    '{username_id}' AND friend_user_id = '{friend_username_id}') OR (user_id =
    '{friend_username_id}' AND friend_user_id = '{username_id}'
    (cursor.execute(query

```

```

        ()result = cursor.fetchone

        ()cursor.close
        ()connection.close

    Return True if they are friends, False otherwise #
        'return result and result[0] == 'accepted

:(def is_active_request(username, friend_username
    Assuming you have a MySQL database connection #
    Replace 'your_database', 'your_user', 'your_password' with your actual #
        database credentials

    (username_id = get_id_from_username(username
    (friend_username_id = get_id_from_username(friend_username
        ()connection = connect_to_kevindb

        ()cursor = connection.cursor

    Check if the users are already friends #
    query = f"SELECT friendship_status FROM friends WHERE (user_id =
    '{username_id}' AND friend_user_id = '{friend_username_id}') OR (user_id =
    '{friend_username_id}' AND friend_user_id = '{username_id}'
        (cursor.execute(query
        ()result = cursor.fetchone

        ()cursor.close
        ()connection.close

```

```

        Return True if they are pending, False otherwise #
        'return result and result[0] == 'pending

    :(def send_friend_request(username, friend_username
    Assuming you have a MySQL database connection #
    Replace 'your_database', 'your_user', 'your_password' with your actual #
        database credentials

    (username_id = get_id_from_username(username
    (friend_username_id = get_id_from_username(friend_username
        )connection = connect_to_kevindb

    )cursor = connection.cursor

    Check if a friend request already exists #
    query = f"SELECT id FROM friends WHERE user_id = '{username_id}' AND
        ""friend_user_id = '{friend_username_id}' AND friendship_status = 'pending
        (cursor.execute(query
        )existing_request = cursor.fetchone

    :if existing_request
    (".print("Friend request already sent
    :else

    Send a new friend request #
    insert_query = f"INSERT INTO friends (user_id, friend_user_id,
    ("friendship_status) VALUES ('{username_id}', '{friend_username_id}', 'pending
        (cursor.execute(insert_query
        )connection.commit
    (".print("Friend request sent successfully

```



```

        ()cursor.close
        ()connection.close

:(def handle_friend_request(username, friend_username, accept

    (username_id = get_id_from_username(username
(friend_username_id = get_id_from_username(friend_username
        ()connection = connect_to_kevinadb

        ()cursor = connection.cursor

        Check if the friend request exists #
query = f"SELECT id FROM friends WHERE user_id = '{friend_username_id}'
    "AND friend_user_id = '{username_id}' AND friendship_status = 'pending
        (cursor.execute(query
        ()request_id = cursor.fetchone

        :if request_id

        Update the friendship status based on the 'accept' parameter #
        'new_status = 'accepted' if accept else 'rejected
update_query = f"UPDATE friends SET friendship_status = '{new_status}'
    "[WHERE id = {request_id[0
        (cursor.execute(update_query
        ()connection.commit

        :else

        ("print("Friend request not found

```

```

        ()cursor.close
    ()connection.close

    :(def remove_friend(username, friend_username
Assuming you have a MySQL database connection #
Replace 'your_database', 'your_user', 'your_password' with your actual #
database credentials

    (username_id = get_id_from_username(username
    (friend_username_id = get_id_from_username(friend_username
    ()connection = connect_to_kevindb

    ()cursor = connection.cursor

    Check if the friendship exists #
    query = f"SELECT id FROM friends WHERE (user_id = '{username_id}' AND
    friend_user_id = '{friend_username_id}') OR (user_id = '{friend_username_id}'
    "AND friend_username_id = '{username_id}') AND friendship_status = 'accepted
    (cursor.execute(query
    ()friendship_id = cursor.fetchone

    :if friendship_id

    Delete the row corresponding to the friendship #
    "[delete_query = f"DELETE FROM friends WHERE id = {friendship_id[0
    (cursor.execute(delete_query
    ()connection.commit
    ("print("Friend removed successfully
    :else
    ("print("Friendship not found

```

```

        ()cursor.close
        ()connection.close

    :(def get_friend_requests(username
        Assuming you have a MySQL database connection #
        Replace 'your_database', 'your_user', 'your_password' with your actual #
        database credentials
        (username_id = get_id_from_username(username
            ()connection = connect_to_kevindb

            ()cursor = connection.cursor

            Retrieve friend requests for the given username #
            query = f"SELECT user_id FROM friends WHERE friend_user_id =
                ""{username_id}" AND friendship_status = 'pending
                (cursor.execute(query
                ()friend_requests = cursor.fetchall

                ()cursor.close
                ()connection.close

            Extract the usernames from the result #
            [friend_requests_list_id = [request[0] for request in friend_requests
                [] = friend_requests_list
                :for friend_id in friend_requests_list_id
                ((friend_requests_list.append(get_username_from_id(friend_id
                    return friend_requests_list

```

```

                                : (def get_user_friends(username
Assuming you have a MySQL database connection #
Replace 'your_database', 'your_user', 'your_password' with your actual #
                                database credentials

                                (username_id = get_id_from_username(username
                                ()connection = connect_to_kevindb

                                ()cursor = connection.cursor

Retrieve friends for the given username #
                                """" = query
                                SELECT
                                CASE
                                WHEN friend_user_id = ? THEN user_id
                                ELSE friend_user_id
                                END AS user_or_friend_id
                                FROM friends
                                WHERE
                                (? = user_id = ? OR friend_user_id)
                                ;'AND friendship_status = 'accepted
                                """"

((cursor.execute(query, (username_id, username_id, username_id
                                ()friends = cursor.fetchall

                                ()cursor.close
                                ()connection.close

```

```

        Extract the friend usernames from the result #
        [friends_list_of_id = [friend[0] for friend in friends
                                ] = friends_list
        :for friend_id in friends_list_of_id
        ((friends_list.append(get_username_from_id(friend_id
                                                    return friends_list

```

```

        : (def add_chat_to_user(username, new_chat_name
                                :try
Connect to your MySQL database (replace with your own connection #
                                (details
                                ()connection = connect_to_kevindb

```

```

        Create a cursor object to interact with the database #
        ()cursor = connection.cursor

```

```

        Retrieve the current chats_list for the user #
        cursor.execute("SELECT chats_list FROM sign_up_table WHERE
                        ((,username = ?", (username
                        ()result = cursor.fetchone

```

```

                                :if result
                                [current_chats_list_json = result[0

        If the current_chats_list_json is None, set it to an empty list #
        current_chats_list = json.loads(current_chats_list_json) if
                                [] current_chats_list_json else
                                :(")")if not new_chat_name.startswith

```

```

(new_chat_name_id = get_id_from_username(new_chat_name
    (current_chats_list.append(new_chat_name_id
                                :else
    (current_chats_list.append(new_chat_name

    Convert the updated_chats_list to JSON format #
    (updated_chats_list_json = json.dumps(current_chats_list

    Update the chats_list for the user #
    cursor.execute("UPDATE sign_up_table SET chats_list = ? WHERE
                                ,"? = username
    ((updated_chats_list_json, username)

    Commit the changes #
    ()connection.commit

    ("'{print(f"Added '{new_chat_name}' to the chats_list for user '{username
                                :else
    ("'{print(f"No user found with username '{username

                                :except sqlite3.Error as err
    ("'{print(f"Error: {err

                                :finally

    Close the cursor and connection #
                                :if cursor
    ()cursor.close
                                :if connection
    ()connection.close

```

```

:(def update_group_image(group_id, image_bytes
                                :try
Establish a connection to the MySQL database #
                                ()connection = connect_to_kevindb
                                ()cursor = connection.cursor

                                "table_name = "my_groups

get_path = f"SELECT group_image_path FROM {table_name} WHERE
                                "? = group_id

Execute the SELECT statement with the parameterized value #
                                ((,cursor.execute(get_path, (group_id

                                Fetch the result #
                                ()result = cursor.fetchone

                                :if result
image_path = result[0] # Extract the file path from the tuple
                                :if image_path
                                (os.remove(image_path

                                :if image_bytes is None
                                group_pic_path = None
                                :else
                                folder_path = files_folder_path
                                (file_name = generate_random_filename(24

```

```

(group_pic_path = os.path.join(folder_path, file_name
    (save_bytes_to_file(image_bytes, group_pic_path
        ("print("saved bytes to image

        Prepare the UPDATE query #
        """ = update_query
        UPDATE my_groups
        ? = SET group_image_path
        ? = WHERE group_id
        """

        Execute the query #
        ((cursor.execute(update_query, (group_pic_path, group_id
            ()connection.commit

        ("{print(f"Group image updated successfully for group ID: {group_id

        :except Exception as e
        ("{print(f"Error updating group image: {e

        :(def get_group_image_by_id(group_id
            :try

        Establish a connection to the MySQL database #
        ()connection = connect_to_kevindb

        """ = select_query
        SELECT group_image_path

```



```

        FROM my_groups
        ? = WHERE group_id
        """

    Execute the query #
    ()cursor = connection.cursor
    ((,cursor.execute(select_query, (group_id
    ()result = cursor.fetchone

        :if result
        [path = result[0
        :if path is None
        return None
        :else
    (image_bytes = file_to_bytes(path
    return image_bytes
        :except Exception as e
    ("{print(f"Error getting group image: {e

        :finally

    Close the database connection #

    ()cursor.close
    ()connection.close

:(def get_user_chats(username
    :try

```

```

        Connect to your MySQL database #
        ()connection = connect_to_kevindb

    Create a cursor object to interact with the database #
        ()cursor = connection.cursor

    Retrieve the current chats_list for the user #
    cursor.execute("SELECT chats_list FROM sign_up_table WHERE
                    ((,username = ?", (username
                    ()result = cursor.fetchone

    If the result is None or empty, return an empty list #
        :[if not result or not result[0
            [] return

    Convert the chats_list JSON to a Python list #
        ([current_chats_list = json.loads(result[0
            [] = chat_list_names
        :for chat in current_chats_list
            :(if isinstance(chat, int
                ((chat_list_names.append(get_username_from_id(chat
                    :else
                        (chat_list_names.append(chat

    (sorted_chats_list = sort_chat_list(chat_list_names, username
        return sorted_chats_list

    :except sqlite3.Error as err
        ("{print(f"Error: {err

```

```

        [] return

    :finally
        Close the cursor and connection #
        :if cursor
            ()cursor.close
        :if connection
            ()connection.close

:(def sort_chat_list(chats_list, username
    Connect to the MySQL database #
    ()conn = connect_to_kevindb

    ()cursor = conn.cursor

    :try
        Iterate over each chat in the chat list #
        {} = chat_timestamps
        :(for index, chat in enumerate(chats_list
Retrieve the timestamp of the last message in the conversation #
            :(")")if chat.startswith
                (group_id = gets_group_attributes_from_format(chat ,_
                    receiver_id = int(group_id) * -1
                    """)cursor.execute
                SELECT MAX(timestamp) AS last_message_timestamp
                    FROM messages
                    ? = WHERE sender_id = ? OR receiver_id

```

```

        ((receiver_id, receiver_id), "")
        :else
        (username_id = get_id_from_username(username
        (chat_id = get_id_from_username(chat
        ""))cursor.execute
        SELECT MAX(timestamp) AS last_message_timestamp
        FROM messages
        WHERE (sender_id = ? AND receiver_id = ?) OR (sender_id = ?
        (? = AND receiver_id
        ((chat_id, username_id, username_id, chat_id), ""
        )row = cursor.fetchone
        :[if row and row[0]
last_message_timestamp = datetime.fromisoformat(row[0]) # Convert
        SQLite timestamp to datetime

        :else
        If no message is found, set last_message_timestamp to a default #
        value
        last_message_timestamp = datetime(1970, 1, 1) # Or any other
        default value you prefer

        chat_timestamps[chat] = last_message_timestamp

        Sort the chat list based on the timestamp of the last message in each #
        conversation
        sorted_chats = sorted(chat_timestamps, key=chat_timestamps.get,
        (reverse=True

        return sorted_chats
    except sqlite3.Error as e

```

```

        Handle any errors that occur during the execution of the query #
            (print("Error sorting chat list:", e
                    return None
                    :finally
                Close the cursor and connection #
                    ()cursor.close
                    ()conn.close

:(def remove_chat_from_user(username, chat_to_remove
                    :try
Connect to your MySQL database (replace with your own connection #
                    (details
                    ()connection = connect_to_kevindb

        Create a cursor object to interact with the database #
            ()cursor = connection.cursor

            Retrieve the current chats_list for the user #
cursor.execute("SELECT chats_list FROM sign_up_table WHERE
                ((,username = ?", (username
                ()result = cursor.fetchone

                    :if result
                        [current_chats_list_json = result[0

        If the current_chats_list_json is None, set it to an empty list #
            current_chats_list = json.loads(current_chats_list_json) if
                [] current_chats_list_json else

```

```

        Remove the specified chat from the current_chats_list #
            :(")")if not chat_to_remove.startswith
(chat_to_remove_id = get_id_from_username(chat_to_remove
            :if chat_to_remove_id in current_chats_list
            (current_chats_list.remove(chat_to_remove_id

    Convert the updated_chats_list to JSON format #
    (updated_chats_list_json = json.dumps(current_chats_list

        Update the chats_list for the user #
cursor.execute("UPDATE sign_up_table SET chats_list = ? WHERE
                                                    ,"? = username
                ((updated_chats_list_json, username)

        Commit the changes #
            ()connection.commit

    print(f"Removed '{chat_to_remove}' from the chats_list for user
                                                    (".''{username
                                                    :else
    print(f"Chat '{chat_to_remove}' not found in the chats_list for user
                                                    (".''{username
                                                    :else

        means chat is group #
        ([:group_to_remove_id = int(chat_to_remove.split("")[0])[1
            :for chat in current_chats_list
            :(")")if chat.startswith
            ([:group_id = int(chat.split("")[0])[1

```

```

        :if group_id == group_to_remove_id
            (current_chats_list.remove(chat

(updated_chats_list_json = json.dumps(current_chats_list
cursor.execute("UPDATE sign_up_table SET chats_list = ?
                                     ,"? = WHERE username
                ((updated_chats_list_json, username)

                Commit the changes #
                ()connection.commit
print(f'Removed '{chat}' from the chats_list for user
                                     (".''{username
                                     break
                                     :else
                (".'{print(f'No user found with username '{username

                :except sqlite3.Error as err
                ("'{print(f'Error: {err

                :finally
                Close the cursor and connection #
                :if cursor
                    ()cursor.close
                :if connection
                    ()connection.close

                :(def get_blocked_users(username
                :try

```

```

        Connect to the MySQL database #
        ()connection = connect_to_kevindb

        Create a cursor object #
        ()cursor = connection.cursor

        Retrieve the blocked_list for the user #
        cursor.execute("SELECT blocked_list FROM sign_up_table WHERE
                        ((,username = ?", (username
                        ()result = cursor.fetchone

        If no result or blocked_list is empty, return an empty list #
        :if not result or not result[0]
            [] return

        Convert the blocked_list JSON to a Python list #
        ([blocked_users_id = json.loads(result[0]
            [] = blocked_users
            :for user_id in blocked_users_id
            ((blocked_users.append(get_username_from_id(user_id

        return blocked_users

    :except sqlite3.Error as err
        ("{print(f"Error: {err
            [] return

    :finally

    Close the cursor and connection #

```



```

                                :if cursor
                                ()cursor.close
                                :if connection
                                ()connection.close

:(def block_user(username, user_to_block
                                :try
                                Connect to the MySQL database #
                                ()connection = connect_to_kevindb

                                Create a cursor object #
                                ()cursor = connection.cursor

                                Retrieve the current blocked_list for the user #
                                cursor.execute("SELECT blocked_list FROM sign_up_table WHERE
                                                ((,username = ?", (username
                                                ()result = cursor.fetchone

                                :if result
                                [blocked_list_json = result[0

                                If the blocked_list_json is None, set it to an empty list #
                                [] blocked_list = json.loads(blocked_list_json) if blocked_list_json else

                                Add the user_to_block to the blocked_list #
                                (user_to_block_id = get_id_from_username(user_to_block
                                (blocked_list.append(user_to_block_id

```

```

        Convert the updated blocked_list to JSON format #
        (updated_blocked_list_json = json.dumps(blocked_list

                                Update the blocked_list for the user #
cursor.execute("UPDATE sign_up_table SET blocked_list = ? WHERE
                                                ,"? = username

                                ((updated_blocked_list_json, username)

                                Commit the changes #
                                ()connection.commit

                                ('.{print(f"Blocked user '{user_to_block}' for user '{username
                                :else

                                ('.{print(f"No user found with username '{username

                                :except sqlite3.Error as err
                                ("){print(f"Error: {err

                                :finally

                                Close the cursor and connection #
                                :if cursor
                                ()cursor.close
                                :if connection
                                ()connection.close

                                :(def unblock_user(username, user_to_unblock
                                :try

                                Connect to the MySQL database #

```

```

()connection = connect_to_kevindb

Create a cursor object #
()cursor = connection.cursor

Retrieve the current blocked_list for the user #
cursor.execute("SELECT blocked_list FROM sign_up_table WHERE
                ((,username = ?", (username
                ()result = cursor.fetchone

                :if result
                [blocked_list_json = result[0

If the blocked_list_json is None, set it to an empty list #
[] blocked_list = json.loads(blocked_list_json) if blocked_list_json else

Remove the user_to_unblock from the blocked_list if it exists #
(user_to_unblock_id = get_id_from_username(user_to_unblock
                :if user_to_unblock_id in blocked_list
                (blocked_list.remove(user_to_unblock_id

Convert the updated blocked_list to JSON format #
(updated_blocked_list_json = json.dumps(blocked_list

Update the blocked_list for the user #
cursor.execute("UPDATE sign_up_table SET blocked_list = ? WHERE
                ,"? = username
                ((updated_blocked_list_json, username)

```

```

Commit the changes #
        ()connection.commit

    ("'{print(f"Unblocked user '{user_to_unblock}' for user '{username
                                :else
print(f"User '{user_to_unblock}' not found in the blocked list for user
                                ("'{username

                                :else
        ("'{print(f"No user found with username '{username

                                :except sqlite3.Error as err
                                ("'{print(f"Error: {err

                                :finally
Close the cursor and connection #
                                :if cursor
                                ()cursor.close
                                :if connection
                                ()connection.close

:(def create_group(group_name, group_manager, group_members_list=None
                                "" = new_chat_name
                                group_id = None

                                :try

                                Connect to the SQLite database #
                                ()connection = connect_to_kevindb

```

```

        Create a cursor object to interact with the database #
            ()cursor = connection.cursor

        Convert group_members_list to a JSON-formatted string #
            [] = group_members_id_list
            :for member in group_members_list
((group_members_id_list.append(get_id_from_username(member
        group_members_json = json.dumps(group_members_id_list) if
        group_members_list else None

        Insert the group into the 'my_groups' table #
        (('timestamp = str(datetime.now()).strftime('%Y-%m-%d %H:%M
        cursor.execute("INSERT INTO my_groups (group_name, group_manager,
            ,"(? ,? ,? ,?) group_members_list, creation_date) VALUES
((group_name, group_manager, group_members_json, timestamp)

        (Get the last inserted row id (equivalent to LAST_INSERT_ID() in MySQL #
            group_id = cursor.lastrowid
            "{new_chat_name = f'({group_id}){group_name

        Commit the changes #
            ()connection.commit

        (!print(f'Group '{group_name}' created successfully

        Add the new chat to each member's chat list #
            :if group_members_list
            :for member in group_members_list

```

```

        (add_chat_to_user(member, new_chat_name

                                :except sqlite3.Error as err
                                    ({print(f"Error: {err

                                :finally

        Close the cursor and connection #

                                :if cursor
                                    ()cursor.close
                                :if connection
                                    ()connection.close

        Return the new chat name and group ID #
        return new_chat_name, group_id

:(def change_group_manager(group_id, new_manager
                                :try

        Connect to the MySQL database #
        ()connection = connect_to_kevindb

        Create a cursor object to interact with the database #
        ()cursor = connection.cursor

        Update the manager for the specified group #
        cursor.execute("UPDATE my_groups SET group_manager = ? WHERE
                        ((group_id = ?", (new_manager, group_id

        Commit the changes #

```

```

        ()connection.commit

print(f"Manager for group (ID: {group_id}) changed to '{new_manager}'
      (!successfully

except sqlite3.Error as err
    (f"print(f"Error: {err

finally

Close the cursor and connection #
    :if cursor
        ()cursor.close
    :if connection
        ()connection.close

:(def get_group_name_by_id(group_id
    :try

Connect to the MySQL database #
()connection = connect_to_kevindb

Create a cursor object to interact with the database #
    ()cursor = connection.cursor

Retrieve the group_name for the specified group_id #
cursor.execute("SELECT group_name FROM my_groups WHERE group_id
              ((,= ?", (group_id
    ()group_name_result = cursor.fetchone

```

```

        Check if the group exists #
        :if group_name_result
        [group_name = group_name_result[0
        return group_name
        :else
        (.print(f"Group with ID {group_id} not found
        return None

        :except sqlite3.Error as err
        ("{print(f"Error: {err
        return None

        :finally
        Close the cursor and connection #
        :if cursor
        ()cursor.close
        :if connection
        ()connection.close

:(def remove_group_member(group_id, group_member
        :try

        Connect to the MySQL database #
        ()connection = connect_to_kevindb

        Create a cursor object to interact with the database #
        ()cursor = connection.cursor

```



```

Retrieve the current group_members_list for the specified group_id #
cursor.execute("SELECT group_members_list FROM my_groups WHERE
                ((,group_id = ?", (group_id
                [row = cursor.fetchone())[0
                [] current_members_list = json.loads(row) if row else

```

```

Remove the group member from the list #
(group_member_id = get_id_from_username(group_member
:if group_member_id in current_members_list
(current_members_list.remove(group_member_id

```

```

Update the group_members_list for the specified group_id #
cursor.execute("UPDATE my_groups SET group_members_list = ?
                ,"? = WHERE group_id
                ((json.dumps(current_members_list), group_id)

```

```

Commit the changes #
(connection.commit

```

```

print(f'Group member '{group_member}' removed from group (ID:
      ("'{group_id}') successfully
      :else

```

```

print(f'Group member '{group_member}' not found in group (ID:
      ("'{group_id}'). No changes made

```

```

:except sqlite3.Error as err
      ("'{print(f'Error: {err

```

```

:finally

```

```

Close the cursor and connection #

```

```

        :if cursor
        ()cursor.close
        :if connection
        ()connection.close

:(def get_group_members(group_id
                          :try
                          Connect to the MySQL database #
                          ()connection = connect_to_kevindb

                          Create a cursor object to interact with the database #
                          ()cursor = connection.cursor

                          Retrieve the group_members_list for the specified group_id #
cursor.execute("SELECT group_members_list FROM my_groups WHERE
                ((,group_id = ?", (group_id
                ()members_list_json = cursor.fetchone

                Check if the group exists and has members #
                :if members_list_json
                ([members_list_id = json.loads(members_list_json[0
                [] = members_list
                :for id in members_list_id
                ((members_list.append(get_username_from_id(id
                return members_list
                :else
                ("print(f"Group with ID {group_id} not found or has no members
                [] return

```

```

        :except sqlite3.Error as err
            ("{print(f'Error: {err
                [] return

            :finally

        Close the cursor and connection #

        :if cursor
            ()cursor.close
        :if connection
            ()connection.close

:(def append_group_member(group_id, group_member
    :try

        Connect to the MySQL database #
        ()connection = connect_to_kevindb

        Create a cursor object to interact with the database #
        ()cursor = connection.cursor

        Retrieve the current group_members_list for the specified group_id #
        cursor.execute("SELECT group_members_list FROM my_groups WHERE
                        ((,group_id = ?", (group_id
                        ()row = cursor.fetchone
        [] current_members_list = json.loads(row[0]) if row else

        Append the new group member to the list #
        (group_member_id = get_id_from_username(group_member

```

```

        (current_members_list.append(group_member_id

Update the group_members_list for the specified group_id #
cursor.execute("UPDATE my_groups SET group_members_list = ? WHERE
                                                         ,"? = group_id
                ((json.dumps(current_members_list), group_id)

Commit the changes #
                ()connection.commit

print(f'Group member '{group_member}' appended to group (ID: {group_id})
                                                         ("!successfully

except sqlite3.Error as err
                ("{print(f'Error: {err

                :finally

Close the cursor and connection #
                :if cursor
                ()cursor.close
                :if connection
                ()connection.close

:(def rename_group(group_id, new_group_name
                :try

Connect to the MySQL database #
                ()connection = connect_to_kevindb

```

```

        Create a cursor object to interact with the database #
            ()cursor = connection.cursor
        "{new_group_name_format = f"({str(group_id)}){new_group_name
            Update the group_name for the specified group_id #
        cursor.execute("UPDATE my_groups SET group_name = ? WHERE
                                ,"? = group_id
                                ((new_group_name_format, group_id)

        Commit the changes #
            ()connection.commit

        print(f"Group (ID: {group_id}) renamed to '{new_group_name}'
                                (!successfully

        :except sqlite3.Error as err
            ("{print(f"Error: {err

        :finally

        Close the cursor and connection #
            :if cursor
                ()cursor.close
            :if connection
                ()connection.close

        :(def get_user_groups(username
            :try

        Connect to the MySQL database #
        ()connection = connect_to_kevindb

```

```

Create a cursor object to interact with the database #
        cursor = connection.cursor

Retrieve the groups where the specified username is a group member #
        cursor.execute("SELECT group_id, group_name, group_members_list,
                        ",group_manager, creation_date
                        ("group_image_path FROM my_groups"

                                [] = user_groups
                                :()for row in cursor.fetchall

                                [] group_members_id_list = json.loads(row[2]) if row[2] else
                                [] = group_members_list
                                :for member_id in group_members_id_list
                                ((group_members_list.append(get_username_from_id(member_id
                                group_image_bytes = file_to_bytes(row[5]) if row[5] else None
Check if the specified username is a group member or the manager #
                                :[if row[4
                                ([timestamp_datetime = datetime.fromisoformat(row[4

                                :[if username in group_members_list or username == row[3
                                })user_groups.append
                                ,[group_id": row[0"
                                ,[group_name": row[1"
                                ,group_members": group_members_list"
                                ,[group_manager": row[3"
                                creation_date": timestamp_datetime.strftime("%Y-%m-%d") if row[4]"
                                else None, # Format the date if not None

```

```

        group_b64_encoded_image":"
        )base64.b64encode(group_image_bytes).decode
utf-8") if group_image_bytes else None"

        ({

        return user_groups

    :except sqlite3.Error as err
        ("){print(f"Error: {err
        [] return

        :finally
        Close the cursor and connection #
        :if cursor
        ()cursor.close
        :if connection
        ()connection.close

    :(def get_group_by_id(group_id
        :try
        Connect to the MySQL database #
        ()connection = connect_to_kevindb

        Create a cursor object to interact with the database #
        ()cursor = connection.cursor

        Retrieve the group with the specified group_id #

```

```

        cursor.execute("SELECT group_id, group_name, group_members_list,
                        ",group_manager, creation_date
group_image_path FROM my_groups WHERE group_id = ?",
                        ((group_id

()group_data = cursor.fetchone

:if group_data
group_members_id_list = json.loads(group_data[2]) if group_data[2] else
[]

[] = group_members_list

:for chat_id in group_members_id_list
((group_members_list.append(get_username_from_id(chat_id
group_image_bytes = file_to_bytes(group_data[5]) if group_data[5] else
None

(timestamp_datetime = datetime.fromisoformat(group_data[4

} = group_info
,[group_id": group_data[0"
,[group_name": group_data[1"
,[group_members": group_members_list"
,[group_manager": group_data[3"
creation_date": timestamp_datetime.strftime("%Y-%m-%d") if"
,[group_data[4] else None
group_b64_encoded_image":
(base64.b64encode(group_image_bytes).decode
utf-8") if group_image_bytes else None"

{

return group_info

```



```

        :else
        return None

    :except sqlite3.Error as err
        ("{print(f"Error: {err
        return None

    :finally
        Close the cursor and connection #
        :if cursor
            ()cursor.close
        :if connection
            ()connection.close

:()def create_database
    :try
(Connect to the SQLite database (or create it if it doesn't exist #
    ('connection = sqlite3.connect('connectify_db.sqlite
        ()connection.commit
        ()connection.close
    ("!print("Database 'connectify_db' created successfully
        :except sqlite3.Error as error
        ("{print(f"Error creating database: {error

:()def create_messages_table
    :try

```

```

        Establish a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        (Execute the SQL code to create the table (SQLite syntax #
            """ = create_table_query
        ) CREATE TABLE IF NOT EXISTS messages
        , message_id INTEGER PRIMARY KEY AUTOINCREMENT
            ,sender_id INTEGER
            ,receiver_id INTEGER
            ,message_content TEXT
            ,message_content_path TEXT
        ,timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            ,type TEXT
            file_name TEXT
            (
            """

        (cursor.execute(create_table_query
        (".print("Table 'messages' created successfully

        :except sqlite3.Error as err
            ("{print(f"Error: {err

        :finally

        Close the cursor and connection #
        :if cursor is not None

```

```

        ()cursor.close
    :if connection is not None
        ()connection.close
    (").print("Connection closed

:()def create_friends_table
    :try

        Establish a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        Execute the SQL code to create the table #
        """ = create_table_query
        ) CREATE TABLE friends
        ,id INTEGER PRIMARY KEY AUTOINCREMENT
        ,(user_id VARCHAR(255
        ,(friend_user_id VARCHAR(255
        friendship_status TEXT CHECK(friendship_status IN ('pending',
        (('accepted', 'rejected
        ;(
        """

        (cursor.execute(create_table_query
        (").print("Table 'friends' created successfully

    :except sqlite3.Error as err
        ("{print(f"Error: {err

```

```

:finally
    Close the cursor and connection #
        :if cursor is not None
            ()cursor.close
        :if connection is not None
            ()connection.close
        (").print("Connection closed

:()def create_groups_table
    :try
        Establish a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        Execute the SQL code to create the table #
        """ = create_table_query

    ) CREATE TABLE IF NOT EXISTS my_groups
    , group_id INTEGER PRIMARY KEY AUTOINCREMENT
        ,(group_name VARCHAR(255
        ,(group_manager VARCHAR(255
,creation_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        ,group_members_list TEXT
        (group_image_path VARCHAR(255
        (

```

```

        """

        (cursor.execute(create_table_query
        (".print("Table 'my_groups' created successfully

        :except sqlite3.Error as err
        ("{print(f"Error: {err

        :finally

        Close the cursor and connection #
        :if cursor is not None
        ()cursor.close
        :if connection is not None
        ()connection.close
        (".print("Connection closed

        :()def create_settings_table
        :try

        Establish a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        Execute the SQL code to create the table #
        """ = create_table_query
    ) CREATE TABLE IF NOT EXISTS settings_table
        ,user_id INTEGER

```

```

        ,volume INTEGER
    ,(output_device VARCHAR(255
    ,(input_device VARCHAR(255
    ,camera_device_index INTEGER
    ,font_size INTEGER
    ,(font VARCHAR(255
    ,(theme_color VARCHAR(255
    ,((censor_data INTEGER CHECK(two_factor_auth IN (0, 1
    ,((private_account INTEGER CHECK(two_factor_auth IN (0, 1
    ,(push_to_talk_bind VARCHAR(255
    ((two_factor_auth INTEGER CHECK(two_factor_auth IN (0, 1
        (
        ""

    (cursor.execute(create_table_query
    (").print("Table 'settings_table' created successfully

    :except sqlite3.Error as err
        ("{print(f"Error: {err

    :finally

    Close the cursor and connection #
        :if cursor is not None
            ()cursor.close
        :if connection is not None
            ()connection.close
    (").print("Connection closed

```

```

:()def create_sign_up_table
    :try
        Establish a connection #
        ()connection = connect_to_kevindb

        Create a cursor #
        ()cursor = connection.cursor

        Execute the SQL code to create the table #
        """ = create_table_query
    ) CREATE TABLE IF NOT EXISTS sign_up_table
    ,id INTEGER PRIMARY KEY AUTOINCREMENT
    ,username VARCHAR(255
    ,password VARCHAR(255
    ,email VARCHAR(255
    ,salt VARCHAR(255
    ,(security_token VARCHAR(255
    ,chats_list TEXT
    ,(profile_pic_path VARCHAR(255
    blocked_list TEXT
    (
        """
    (cursor.execute(create_table_query
    (").print("Table 'sign_up_table' created successfully

    :except sqlite3.Error as err
        ("){print(f"Error: {err

```

```

:finally
    Close the cursor and connection #
        :if cursor is not None
            ()cursor.close
        :if connection is not None
            ()connection.close
        (").print("Connection closed

:()def create_songs_table
    :try
        Establish a connection #
        ()connection = connect_to_kevinadb

        Create a cursor #
        ()cursor = connection.cursor
        """ = create_table_query

    ) CREATE TABLE IF NOT EXISTS songs
    ,id INTEGER PRIMARY KEY AUTOINCREMENT
    ,title VARCHAR(255) NOT NULL
    ,mp3_file_path VARCHAR(255) NOT NULL
    ,(thumbnail_path VARCHAR(255)
    ,owner_id INTEGER
    ,duration VARCHAR(255) NOT NULL
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP

    (
        """

    (cursor.execute(create_table_query

```



```

        ("print("Table 'songs' created successfully
                                :except sqlite3.Error as err
        ("print(f"Error in creating songs table: {err

```

```

                                :finally

```

```

        Close the cursor and connection #

```

```

                                :if cursor is not None

```

```

                                    ()cursor.close

```

```

                                :if connection is not None

```

```

                                    ()connection.close

```

```

        ("print("Connection closed

```

```

                                :()def create_tables

```

```

                                    ()create_sign_up_table

```

```

                                    ()create_groups_table

```

```

                                    ()create_friends_table

```

```

                                    ()create_messages_table

```

```

                                    ()create_settings_table

```

```

                                    ()create_songs_table

```

```

                                :(def clear_sqlite_table(table_name

```

```

                                    :try

```

```

        Connect to the SQLite database #

```

```

                                ()conn = connect_to_kevindb

```

```

                                ()cursor = conn.cursor

```

```

Construct and execute the DELETE statement #
"{delete_query = f"DELETE FROM {table_name
    (cursor.execute(delete_query

Commit the transaction #
    ()conn.commit
("{print(f"All rows deleted from table '{table_name

except sqlite3.Error as e
    ("{print(f"Error occurred: {e

finally

Close the connection #
    :if conn
        ()conn.close

:()def connect_to_kevindb
    :try
        Connect to the existing SQLite database #
        ('connection = sqlite3.connect('connectify_db.sqlite
            return connection
        :except sqlite3.Error as error
            ("{print(f"Error connecting to database: {error
                return None

        ()create_database

```

```

                                ()create_tables

                                import os
                                import smtplib

                                from email.mime.multipart import MIMEMultipart
                                from email.mime.text import MIMEText
                                from email.mime.image import MIMEImage
                                import ssl

                                ('email_password = os.environ.get('email_password
                                "from_email = "appmails742@gmail.com

:(def send_confirmation_to_client_email(receiver_mail, account_name
    'logo_path = 'discord_app_assets/connectify_icon.png
    connectify_account = account_name
    password = email_password
    to_email = receiver_mail
    "subject = "Welcome to connectify
    ""body = f
    ,{Hi {connectify_account
Welcome to Connectify. Your new account comes with access to Connectify
    .products, apps, and services
    ""

                                ()em = MIMEMultipart
                                em['From'] = from_email
                                em['To'] = to_email
                                em['Subject'] = subject

```

```

        ('body_text = MIMEText(body, 'plain
            (em.attach(body_text

    ()context = ssl.create_default_context

    :with open(logo_path, 'rb') as logo_file
        ()logo_content = logo_file.read
    ('logo_attachment = MIMEImage(logo_content, name='logo.png
        (em.attach(logo_attachment

    :with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp
        (smtp.login(from_email, password
        ((smtp.sendmail(from_email, to_email, em.as_string

    :
    (def send_login_code_to_client_email(code, receiver_mail, account_name
        'logo_path = 'discord_app_assets/connectify_icon.png
    password = email_password # Ensure you have email_password defined
                                somewhere
        to_email = receiver_mail
        "subject = "Login Verification Code for Connectify
        ""body = f
            <p>
                <Dear {account_name},<br><br>
                    We received a login request for your Connectify Account. Your verification
                        <code is:<br><br>
                            <strong>{code}</strong><br><br>

```

If you did not initiate this login attempt - It means someone else have
.access to your Connectify password
We recommend for you to change your password As soon as possible,
<please ignore this email.

<Sincerely yours,

The Connectify Team

<p/>

<-- img src="cid:logo"> <!-- This references the inline image>
""""

```
()em = MIMEMultipart
em['From'] = from_email
em['To'] = to_email
em['Subject'] = subject
```

```
Attach HTML body #
('body_html = MIMEText(body, 'html
(em.attach(body_html
```

```
Attach inline logo #
:with open(logo_path, 'rb') as logo_file
('logo_attachment = MIMEImage(logo_file.read(), name='logo.png
('<logo_attachment.add_header('Content-ID', '<logo
(em.attach(logo_attachment
```

```
()context = ssl.create_default_context
```

```
:with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp
    (smtp.login(from_email, password
    (smtp.sendmail(from_email, to_email, em.as_string
```

```
:(def send_sing_up_code_to_client_email(code, receiver_mail, account_name
    'logo_path = 'discord_app_assets/connectify_icon.png
    connectify_account = account_name
    password = email_password
    to_email = receiver_mail
    "subject = "Verification code for Connectify
    """"body = f
    <p>
    <Dear {to_email},<br><br>
```

We received a request to create your Connectify Account through our
<website. Your Connectify verification code is:

{code}

If you did not request this code, it is possible that someone else is trying to
create your Connectify Account - {connectify_account}. Do not forward or give
<this code to anyone.

You received this message because this email address is listed as the email
to create your Connectify Account - {connectify_account}. If that is incorrect,
<please visit our site to remove your email from a Connectify Account.

<Sincerely yours,

The Connectify Accounts team

<p/>

<-- img src="cid:logo"> <!-- This references the inline image>

""

()em = MIMEMultipart

em['From'] = from_email

em['To'] = to_email

em['Subject'] = subject

Attach HTML body #

('body_html = MIMEText(body, 'html

(em.attach(body_html

Attach inline logo #

:with open(logo_path, 'rb') as logo_file

('logo_attachment = MIMEImage(logo_file.read(), name='logo.png

('<logo_attachment.add_header('Content-ID', '<logo

(em.attach(logo_attachment

()context = ssl.create_default_context

:with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp

(smtp.login(from_email, password

((smtp.sendmail(from_email, to_email, em.as_string

```

:(def send_forget_password_code_to_email(code, receiver_mail, account_name
      'logo_path = 'discord_app_assets/connectify_icon.png
      connectify_account = account_name
      password = email_password
      to_email = receiver_mail
      "subject = "Verification code for Connectify
      """"body = f
          <p>
          <Dear {to_email},<br><br>

```

```

We received a request to reset your Connectify Account's Password through
<our website. Your Connectify verification code is:<br><br>

```

```

<strong>{code}</strong><br><br>

```

```

If you did not request this code, it is possible that someone else is trying to
change your Connectify Account Password- {connectify_account}. Do not
<forward or give this code to anyone.<br><br>

```

```

You received this message because this email address is listed as the email
of your Connectify Account - {connectify_account}. If that is incorrect, please visit
<our site to remove your email from a Connectify Account.<br><br>

```

```

<Sincerely yours,<br><br>

```

```

The Connectify Accounts team

```

```

<p/>

```

```

<-- img src="cid:logo"> <!-- This references the inline image>

```

```

""""

```



```

        em = MIMEMultipart
        em['From'] = from_email
        em['To'] = to_email
        em['Subject'] = subject

        Attach HTML body #
        ('body_html = MIMEText(body, 'html
        (em.attach(body_html

        Attach inline logo #
        :with open(logo_path, 'rb') as logo_file
        ('logo_attachment = MIMEImage(logo_file.read(), name='logo.png
        ('<logo_attachment.add_header('Content-ID', '<logo
        (em.attach(logo_attachment

        ()context = ssl.create_default_context

        :with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp
        (smtp.login(from_email, password
        ((smtp.sendmail(from_email, to_email, em.as_string

        :
        (def send_changed_password_to_email(receiver_mail, account_name
        'logo_path = 'discord_app_assets/connectify_icon.png
        connectify_account = account_name
        password = email_password
        to_email = receiver_mail
        "!"subject = "Welcome to connectify

```

```

        """body = f
        ,{Hello {connectify_account
You have successfully changed your Connectify account password. If you
did not make this request, please reset the passwords of your email address and
        .Connectfiy account
        ,Thank you
        The Connectify Team
        """

```

```

        ()em = MIMEMultipart
        em['From'] = from_email
        em['To'] = to_email
        em['Subject'] = subject

        ('body_text = MIMEText(body, 'plain
        (em.attach(body_text

        ()context = ssl.create_default_context

        :with open(logo_path, 'rb') as logo_file
        ()logo_content = logo_file.read
        ('logo_attachment = MIMEImage(logo_content, name='logo.png
        (em.attach(logo_attachment

        :with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp
        (smtp.login(from_email, password
        ((smtp.sendmail(from_email, to_email, em.as_string
        :Example usage #

```

```

        * from PyQt5.QtWidgets import
        from PyQt5.QtGui import QColor
from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit
        from PyQt5.QtCore import QSize
        from PyQt5.QtGui import QIcon, QPixmap

        :(def filter_and_sort_chats(search_str, chat_list
        Check if chat_list is a list of tuples or a list of strings #
            :if len(chat_list) == 0
                [] return
            :(if isinstance(chat_list[0], tuple
Filter out tuples where chat_name does not contain the search_str #
                :if not search_str
                    return chat_list
        filtered_chats = [(chat_name, unread_messages) for chat_name,
                        unread_messages in chat_list if
                        [()]search_str.lower() in chat_name.lower

        Sort the filtered_chats based on relevance to the search_str #
            ,sorted_chats = sorted(filtered_chats
key=lambda x: (not x[0].lower().startswith(search_str.lower())),
                        (((x[0].lower
                return sorted_chats
            :(elif isinstance(chat_list[0], str
        Filter out strings that do not contain the search_str #
            :if not search_str
                return chat_list

```

```
filtered_chats = [chat_name for chat_name in chat_list if search_str.lower()
                  [()]in chat_name.lower
```

```

Sort the filtered_chats based on relevance to the search_str #
    ,sorted_chats = sorted(filtered_chats
key=lambda x: (not x.lower().startswith(search_str.lower())),
                (((()x.lower
                return sorted_chats
                :else
Handle other cases or raise an exception based on your requirements #
    ("raise ValueError("Invalid format for chat_list
```

```

:(class FriendsBox(QWidget
:(def __init__(self, friends_list, requests_list, Network, username, parent=None
                ()__super().__init
                self.font_size = 60
                Styling #
                [] = self.raised_elements

                self.parent = parent
                self.Network = Network
                self.username = username
                self.requests_list = requests_list
                "" = style_sheet
                ;color: white
                ;font-size: 40px
                ;margin-bottom: 10px
                ""
```

```

        "" = main_style_sheet
        ;color: white
        ;font-size: 40px
        ;padding: 10px
        /* border: 2px solid #3498db; /* Blueish border color
        ;border-radius: 10px
        ;margin-bottom: 10px
        ""

        (self.friends_label = QPushButton(" Social", self
        (self.block_friend_label = QLabel(self
        (self.remove_friend_label = QLabel(self
        ""))self.friends_label.setStyleSheet
        ;color: white
        ;font-size: 15px
        /* border: none; /* Remove the border
        ;border-radius: 5px
        ;padding: 5px
        ;margin-bottom: 2px
        /* text-align: left; /* Align the text to the left
        /* alignment: left; /* Align the icon and text to the left
        /* padding-left: 10px; /* Adjust the starting position to the right
        } hover
        /* background-color: transparent; /* Remove hover effect
        {

        (""

```

```

icon = QIcon("discord_app_assets/friends_icon.png") # Replace with the
                                                    path to your icon image

        (self.friends_label.setIcon(icon
                                     button_y = 10

Set the position and size of the button #
        friend_x = 600
        friends_label_y = 0
        (self.friends_label.move(friend_x - 30, button_y
Set the text alignment to show both the icon and text #

Optional: Adjust the spacing between the icon and text #
self.friends_label.setIconSize(QSize(50, 50)) # Adjust size as needed

        """"selecting_buttons_stylesheet = (f
                                                    }} QPushButton
background-color: {self.parent.background_color_hex}; /* Use your
                                                    /* desired blue color
border: 2px solid {self.parent.standard_hover_color}; /* Use a slightly
                                                    /* darker shade for the border
                                                    ;border-radius: 5px
                                                    ;padding: 8px 16px
                                                    ;color: #b9c0c7
                                                    ;font-family: Arial, sans-serif
                                                    ;font-size: 14px
                                                    ;font-weight: normal
;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
                                                    {{

                                                    }} QPushButton:hover

```

```

;{background-color: {self.parent.standard_hover_color
}}

}} QPushButton:pressed
;background-color: #202225
;border-color: #72767d
}}
(""""

"""selecting_button_pressed_stylesheet = (f
}} QPushButton
background-color: #3498db; /* Use your desired color for pressed */ state
border: 2px solid {self.parent.standard_hover_color}; /* Use a slightly
/* darker shade for the border
;border-radius: 5px
;padding: 8px 16px
;color: #b9c0c7
;font-family: Arial, sans-serif
;font-size: 14px
;font-weight: normal
;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
}}
}} QPushButton:hover
;background-color: #2980b9
}}
}} QPushButton:pressed
background-color: #2c3e50; /* Use your desired color for pressed */ state

```

```

border-color: #34495e; /* Use a slightly darker shade for the border in
                                                                    /* pressed state
                                                                    {{
                                                                    (""

border1_width = 725
border1_height = self.friends_label.height() + 48
(self.border1_label = QLabel(self
    ""self.border1_label.setStyleSheet(f
;{border: 2px solid {self.parent.standard_hover_color
    ;border-radius: 5px
    ;padding: 5px
    ;margin-bottom: 2px
    ("
self.border1_label.setGeometry(friend_x - 40, 0, border1_width,
    (border1_height
    ()self.border1_label.lower

border2_width = border1_width
border3_height = self.friends_label.height() + 900
(self.border2_label = QLabel(self
    ""self.border2_label.setStyleSheet(f
;{border: 2px solid {self.parent.standard_hover_color
    ;border-radius: 5px
    ;padding: 5px
    ;margin-bottom: 2px
    ("
:"if self.parent.friends_box_page == "add friend
border2_height = 195 - border1_height + 85

```



```

:else

border2_height = 195 - border1_height + 10
self.border2_label.setGeometry(friend_x - 40, border1_height - 3,
                              (border2_width, border2_height
                              )self.border2_label.lower

(self.border3_label = QLabel(self
    ""self.border3_label.setStyleSheet(f
;{border: 2px solid {self.parent.standard_hover_color
    ;border-radius: 5px
    ;padding: 5px
    ;margin-bottom: 2px
    ("
self.border3_label.setGeometry(friend_x - 40, 195, border2_width,
                              (border3_height
                              )self.border3_label.lower

buttons_y = button_y + 10

online_button_x = friend_x + 150
(self.online_button = QPushButton("Online", self
(self.online_button.move(online_button_x, buttons_y
    self.online_button_height = 37
(self.online_button.setFixedHeight(self.online_button_height
self.online_button.clicked.connect(self.online_button_pressed) # Connect
                                                                the button to the function
(self.online_button.setStyleSheet(selecting_buttons_stylesheet

all_button_x = online_button_x + self.online_button.width() + 10

```

```

        (self.all_button = QPushButton("All", self
        (self.all_button.move(all_button_x, buttons_y
                                self.all_button_height = 37
        (self.all_button.setFixedHeight(self.all_button_height
self.all_button.clicked.connect(self.all_button_pressed) # Connect the
                                button to the function

        (self.all_button.setStyleSheet(selecting_buttons_stylesheet

        pending_button_x = all_button_x + self.all_button.width() - 15
        (self.Pending_button = QPushButton("Pending", self
        (self.Pending_button.move(pending_button_x, buttons_y
                                self.Pending_button_height = 37
        (self.Pending_button.setFixedHeight(self.Pending_button_height
self.Pending_button.clicked.connect(self.pending_button_pressed) #
                                Connect the button to the function

        (self.Pending_button.setStyleSheet(selecting_buttons_stylesheet

        blocked_button_x = pending_button_x + self.Pending_button.width() + 15
        (self.blocked_button = QPushButton("Blocked", self
        (self.blocked_button.move(blocked_button_x, buttons_y
                                self.blocked_button_height = 37
        (self.blocked_button.setFixedHeight(self.blocked_button_height
self.blocked_button.clicked.connect(self.blocked_button_pressed) #
                                Connect the button to the function

        (self.blocked_button.setStyleSheet(selecting_buttons_stylesheet

        add_friend_button_x = blocked_button_x + self.blocked_button.width() + 10
        (self.add_friend = QPushButton("Add Friend", self
        (self.add_friend.move(add_friend_button_x, buttons_y

```

```

        self.add_friend_height = 37
        (self.add_friend.setFixedHeight(self.add_friend_height
self.add_friend.clicked.connect(self.add_friend_button_pressed) # Connect
                                the button to the function
        (self.add_friend.setStyleSheet(selecting_buttons_stylesheet

        search_x = friend_x - 10
        search_y = border1_height + 20
        search_height = 45
        search_width = border2_width - 65
        self.friends_list = friends_list

        (self.search = QLineEdit(self
                                ()self.search.hide

        (self.social_label = QLabel("", self
                                ()self.social_label.hide
        "self.search_box_color = "white
        : "if self.parent.background_color == "Black and White
        "self.search_box_color = "black

        : "if self.parent.friends_box_page == "online

        [] = friends_box_list
                                :try

        :if self.parent.current_friends_box_search
friends_box_list = self.parent.temp_search_list
                                :else
friends_box_list = self.parent.online_users_list

```

```

        except Exception as e
            ({print(f"error in friends_box{e

(self.social_label = QLabel(f"ONLINE — {len(friends_box_list)}", self
self.social_label.setStyleSheet("color: white; font-size: 12px; font-weight:
                                (";bold

        Adjust the position and size of the label as needed #
        (self.social_label.move(search_x, search_y + 60
        self.social_label.adjustSize() # Adjust the size to fit the content

(self.online_button.setStyleSheet(selecting_button_pressed_stylesheet

                                (self.search = QLineEdit(self
                                ("self.search.setPlaceholderText("Search
                                )self.search.setStyleSheet
                                f"background-color: {self.parent.standard_hover_color}; color:
                                  {self.search_box_color}; padding: 10px; border: 1px solid
                                (";{self.parent.standard_hover_color}; border-radius: 5px; font-size: 14px
                                self.search.setGeometry(search_x, search_y, search_width,
                                                        (search_height

(self.search.textChanged.connect(self.on_text_changed_in_contact_search

                                self.default_starting_y = 200
                                [] = self.friend_labels
                                (friend_starter_y = 200 + (self.parent.friends_box_index * -50
                                self.parent.friends_box_index_y_start = friend_starter_y

                                friends_label_x = search_x

```

```

        :for friend in friends_box_list
            (friend_label = QLabel(friend, self
            (friend_label.setStyleSheet(style_sheet
            (friend_label.move(friends_label_x + 25, friend_starter_y
            friend_label.setFixedHeight(self.font_size) # Increase height
            friend_label.adjustSize() # Ensure the label size is adjusted to its
                                                    content

            (line = QFrame(self
            line.setGeometry(friend_x - 40, friend_starter_y + self.font_size + 5,
                                                    (border2_width, 2
            line.setStyleSheet(f"background-color:
            {self.parent.standard_hover_color};") # Set line color

            (chat_button = QPushButton(self
            chat_button_x = 1235
            (self.chat_label = QLabel("Message", self
            (self.raised_elements.append(self.chat_label
            )self.connect_button_label_pair
            ,chat_button
            ,self.chat_label
            ,"Message"
            ,"discord_app_assets/press_chat_icon.png"
            ,self.chat_label
            ,chat_button_x
            ,friend_starter_y + 10
            friend # Pass the friend parameter here
            (

```

```

(remove_friend_button = QPushButton(self
remove_friend_button_x = chat_button_x - 60
(self.remove_friend_label = QLabel("Remove", self
(self.raised_elements.append(self.remove_friend_label
)self.connect_button_label_pair
,remove_friend_button
,self.remove_friend_label
,"Remove"
,"discord_app_assets/remove_friend_icon.png"
,self.remove_friend_label
,remove_friend_button_x
,friend_starter_y + 10
friend # Pass the friend parameter here
(

```

```

(block_friend_button = QPushButton(self
(self.block_friend_label = QLabel("Block", self
(self.raised_elements.append(self.block_friend_label
)self.connect_button_label_pair
,block_friend_button
,self.block_friend_label
,"Block"
,"discord_app_assets/block_icon.png"
,self.block_friend
,remove_friend_button_x - 60
,friend_starter_y + 10
friend # Pass the friend parameter here
(

```

```

(circle_label = QLabel(self
(circle_label.setGeometry(friends_label_x, friend_starter_y + 20, 20, 20
:if friend in self.parent.online_users_list
("self.draw_circle(circle_label, "green
:else
("self.draw_circle(circle_label, "gray
:()if friend_starter_y > self.parent.height
self.parent.is_friends_box_full = True
("print("smaller than 0
break

friend_starter_y += 70
(self.friend_labels.append(friend_label
()self.raise_all_element
:()if friend_starter_y < self.parent.height
self.parent.is_friends_box_full = False

:"if self.parent.friends_box_page == "all
[] = friends_box_list
:try

:if self.parent.current_friends_box_search
friends_box_list = self.parent.temp_search_list
:else
friends_box_list = self.parent.friends_list
:except Exception as e
("{print(f"error in friends_box{e
Friends Label #

```

```

(self.all_button.setStyleSheet(selecting_button_pressed_stylesheet

self.social_label = QLabel(f"ALL FRIENDS — {len(friends_box_list)}",
                                                                    (self
self.social_label.setStyleSheet("color: white; font-size: 12px; font-weight:
                                                                    (";bold

        Adjust the position and size of the label as needed #
        (self.social_label.move(search_x, search_y + 60
self.social_label.adjustSize() # Adjust the size to fit the content

        (self.search = QLineEdit(self
        ("self.search.setPlaceholderText("Search
        )self.search.setStyleSheet
        f"background-color: {self.parent.standard_hover_color}; color:
          {self.search_box_color}; padding: 10px; border: 1px solid
        (";{self.parent.standard_hover_color}; border-radius: 5px; font-size: 14px
        self.search.setGeometry(search_x, search_y, search_width,
                                (search_height

(self.search.textChanged.connect(self.on_text_changed_in_contact_search

        self.default_starting_y = 200
        [] = self.friend_labels
        (friend_starter_y = 200 + (self.parent.friends_box_index * -50
        self.parent.friends_box_index_y_start = friend_starter_y

        friends_label_x = search_x
        :for friend in friends_box_list
        (friend_label = QLabel(friend, self

```



```

        (friend_label.setStyleSheet(style_sheet
        (friend_label.move(friends_label_x + 25, friend_starter_y
        friend_label.setFixedHeight(self.font_size) # Increase height
        friend_label.adjustSize() # Ensure the label size is adjusted to its
content

```

```

        (line = QFrame(self
line.setGeometry(friend_x - 40, friend_starter_y + self.font_size + 5,
        (border2_width, 2
        line.setStyleSheet(f"background-color:
        {self.parent.standard_hover_color};") # Set line color

```

```

        (chat_button = QPushButton(self
        chat_button_x = 1235
        (self.chat_label = QLabel("Message", self
        (self.raised_elements.append(self.chat_label
        )self.connect_button_label_pair
        ,chat_button
        ,self.chat_label
        ,"Message"
        ,"discord_app_assets/press_chat_icon.png"
        ,self.open_chat
        ,chat_button_x
        ,friend_starter_y + 10
        friend # Pass the friend parameter here

```

(

```

        (remove_friend_button = QPushButton(self
        remove_friend_button_x = chat_button_x - 60

```

```

        (self.remove_friend_label = QLabel("Remove", self
(self.raised_elements.append(self.remove_friend_label
                                )self.connect_button_label_pair
                                ,remove_friend_button
                                ,self.remove_friend_label
                                ,"Remove"
                                ,"discord_app_assets/remove_friend_icon.png"
                                ,self.remove_friend
                                ,remove_friend_button_x
                                ,friend_starter_y + 10
                                friend # Pass the friend parameter here
                                (

(block_friend_button = QPushButton(self

        (self.block_friend_label = QLabel("Block", self
(self.raised_elements.append(self.block_friend_label
                                )self.connect_button_label_pair
                                ,block_friend_button
                                ,self.block_friend_label
                                ,"Block"
                                ,"discord_app_assets/block_icon.png"
                                ,self.block_friend
                                ,remove_friend_button_x - 60
                                ,friend_starter_y + 10
                                friend # Pass the friend parameter here
                                (

```

```

(circle_label = QLabel(self
(circle_label.setGeometry(friends_label_x, friend_starter_y + 20, 20, 20
:if friend in self.parent.online_users_list
("self.draw_circle(circle_label, "green
:else
("self.draw_circle(circle_label, "gray
:()if friend_starter_y > self.parent.height
self.parent.is_friends_box_full = True
("print("smaller than 0
break

friend_starter_y += 70
(self.friend_labels.append(friend_label
()self.raise_all_element
:()if friend_starter_y < self.parent.height
self.parent.is_friends_box_full = False

:"if self.parent.friends_box_page == "pending
:try
[] = friends_box_list
:try
:if self.parent.current_friends_box_search
friends_box_list = self.parent.temp_search_list
:else
friends_box_list = self.parent.request_list
:except Exception as e
("{print(f"error in friends_box{e
Friends Label #

```

```

(self.Pending_button.setStyleSheet(selecting_button_pressed_stylesheet

(self.social_label = QLabel(f"Pending — {len(friends_box_list)}", self
    self.social_label.setStyleSheet("color: white; font-size: 12px;
                                    (";font-weight: bold

        Adjust the position and size of the label as needed #
        (self.social_label.move(search_x, search_y + 60
self.social_label.adjustSize() # Adjust the size to fit the content
        [] = self.requests_items

        self.default_starting_y = 200
        [] = self.friend_labels
(request_starter_y = 200 + (self.parent.friends_box_index * -50
    self.parent.friends_box_index_y_start = request_starter_y

        request_x = search_x

        (self.search = QLineEdit(self
            ("self.search.setPlaceholderText("Search
                )self.search.setStyleSheet
f"background-color: {self.parent.standard_hover_color}; color:
    {self.search_box_color}; padding: 10px; border: 1px solid
    (";{self.parent.standard_hover_color}; border-radius: 5px; font-size: 14px
        self.search.setGeometry(search_x, search_y, search_width,
                                (search_height

(self.search.textChanged.connect(self.on_text_changed_in_contact_search

```

```

        :for request in requests_list
            (request_label = QLabel(request, self
            (request_label.setStyleSheet(style_sheet
            (request_label.move(request_x, request_starter_y
            request_label.setFixedHeight(70) # Increase height

V" (Green) Button" #
            (accept_button = QPushButton("✓", self
            "")accept_button.setStyleSheet
/* background-color: #2ecc71; /* Green color
            ;color: white
            ;padding: 10px
            ;border: 2px solid #2ecc71
            ;border-radius: 10px
            ;font-size: 24px
            (""
            (accept_button.move(request_x + 200, request_starter_y
            accept_button.setFixedHeight(50) # Increase height
            accept_button.setFixedWidth(50) # Fixed width

X" (Red) Button" #
            (reject_button = QPushButton("✗", self
            "")reject_button.setStyleSheet
/* background-color: #e74c3c; /* Red color
            ;color: white
            ;padding: 10px
            ;border: 2px solid #e74c3c
            ;border-radius: 10px

```

```

;font-size: 24px
        (""
(reject_button.move(request_x + 260, request_starter_y
reject_button.setFixedHeight(50) # Increase height
reject_button.setFixedWidth(50) # Fixed width

        request_starter_y += 70
        (self.requests_items.append(request_label
        (self.requests_items.append(accept_button
        (self.requests_items.append(reject_button
        :()if request_starter_y > self.parent.height
            self.parent.is_friends_box_full = True
            ("print("smaller then 0
                break
        :()if request_starter_y < self.parent.height
            self.parent.is_friends_box_full = False

        :(for i in range(0, len(self.requests_items), 3
[accept_button = self.requests_items[i + 1
[reject_button = self.requests_items[i + 2

        )accept_button.clicked.connect
lambda checked, index=i: self.handle_friend_request(index,
                                                    ((accept=True
        )reject_button.clicked.connect
lambda checked, index=i: self.handle_friend_request(index,
                                                    ((accept=False
        ()self.raise_all_element
        :except Exception as e

```

```

({print(f"error pending page {e

        : "if self.parent.friends_box_page == "add friend
(self.add_friend.setStyleSheet(selecting_button_pressed_stylesheet
        (self.social_label = QLabel(f"ADD FRIEND", self
self.social_label.setStyleSheet("color: white; font-size: 16px; font-weight:
        (";bold

        Adjust the position and size of the label as needed #
        (self.social_label.move(search_x, search_y + 5
self.social_label.adjustSize() # Adjust the size to fit the content

self.add_friend_label = QLabel(f"You can add friends with their Connectify
        (username.", self
        (";self.add_friend_label.setStyleSheet("color: white; font-size: 12px

        Adjust the position and size of the label as needed #
        (self.add_friend_label.move(search_x, search_y + 55
self.add_friend_label.adjustSize() # Adjust the size to fit the content

        (self.add_friend_entry = QLineEdit(self
        ("self.add_friend_entry.setPlaceholderText("Add Friend
self.add_friend_entry.setGeometry(search_x, search_y + 100,
        (search_width - 10, search_height + 50
        )self.add_friend_entry.setStyleSheet
        f"background-color: {self.parent.standard_hover_color}; color:
        {self.search_box_color}; padding: 10px; border: 1px solid
        (";{self.parent.standard_hover_color}; border-radius: 5px; font-size: 14px
        self.add_friend_entry.setFixedHeight(40) # Increase height

```

```

        : "if self.parent.friends_box_page == "blocked"
            [] = self.friend_labels
        (friend_starter_y = 200 + (self.parent.friends_box_index * -50
            [] = friends_box_list
            : try
            : if self.parent.current_friends_box_search
                friends_box_list = self.parent.temp_search_list
            : else
                friends_box_list = self.parent.blocked_list
            : except Exception as e
                ({print(f"error in friends_box{e
                    Friends Label #
(self.blocked_button.setStyleSheet(selecting_button_pressed_stylesheet

    (self.social_label = QLabel(f"BLOCKED — {len(friends_box_list)}", self
    self.social_label.setStyleSheet("color: white; font-size: 12px; font-weight:
                                                                    ("; bold

    Adjust the position and size of the label as needed #
        (self.social_label.move(search_x, search_y + 60
    self.social_label.adjustSize() # Adjust the size to fit the content

        (self.search = QLineEdit(self
        ("self.search.setPlaceholderText("Search
            )self.search.setStyleSheet
            f"background-color: {self.parent.standard_hover_color}; color:
                {self.search_box_color}; padding: 10px; border: 1px solid
            ("; {self.parent.standard_hover_color}; border-radius: 5px; font-size: 14px
            self.search.setGeometry(search_x, search_y, search_width,
                                                                    (search_height

```



```

(self.search.textChanged.connect(self.on_text_changed_in_contact_search

                                friends_label_x = search_x
                                :for friend in friends_box_list
                                (friend_label = QLabel(friend, self
                                (friend_label.setStyleSheet(style_sheet
                                (friend_label.move(friends_label_x + 25, friend_starter_y
                                friend_label.setFixedHeight(self.font_size) # Increase height
                                friend_label.adjustSize() # Ensure the label size is adjusted to its
                                                                content

                                (line = QFrame(self
                                line.setGeometry(friend_x - 40, friend_starter_y + self.font_size + 5,
                                                                (border2_width, 2
                                line.setStyleSheet(f"background-color:
                                {self.parent.standard_hover_color};") # Set line color

                                unblock_friend_button_x = 1235

                                (unblock_friend_button = QPushButton(self

                                (self.block_friend_label = QLabel("Block", self
                                (self.raised_elements.append(self.block_friend_label
                                                                )self.connect_button_label_pair
                                                                ,unblock_friend_button
                                                                ,self.block_friend_label
                                                                ,"Unblock"
                                                                ,"discord_app_assets/block_icon.png"

```

```

        ,self.unblock_friend
        ,unblock_friend_button_x - 60
        ,friend_starter_y + 10
friend # Pass the friend parameter here
        (

:()if friend_starter_y > self.parent.height
self.parent.is_friends_box_full = True
        ("print("smaller then 0
        break

        friend_starter_y += 70
        (self.friend_labels.append(friend_label
        ()self.raise_all_element
        :()if friend_starter_y < self.parent.height
        self.parent.is_friends_box_full = False

        (self.error_friend = QLabel("couldn't find user", self
        (self.error_friend.move(search_x, search_y + 100 + search_height
        ()self.error_friend.hide
        ("";self.error_friend.setStyleSheet("color: red

        (self.request_sent = QLabel("Request sent", self
        (self.request_sent.move(search_x, search_y + 100 + search_height
        ()self.request_sent.hide
        ("";self.request_sent.setStyleSheet("color: green

        (self.already_friends = QLabel("Already friends", self

```

```

(self.already_friends.move(search_x, search_y + 100 + search_height
                                )self.already_friends.hide
                                ("";self.already_friends.setStyleSheet("color: red

                                (self.already_sent_request = QLabel("Request is pending", self
(self.already_sent_request.move(search_x, search_y + 100 + search_height
                                )self.already_sent_request.hide
                                ("";self.already_sent_request.setStyleSheet("color: gray

def connect_button_label_pair(self, button, label, label_text, icon_path,
                                :(click_callback, x, y, friend
                                (button.setFixedSize(40, 40
                                ((button_icon = QIcon(QPixmap(icon_path
                                (button.setIcon(button_icon
                                : "if label_text != "Message
                                (((button.setIconSize(button_icon.actualSize(QSize(26, 26
                                :else
                                (((button.setIconSize(button_icon.actualSize(QSize(41, 41
                                """"button.setStyleSheet(f
                                }} QPushButton: hover
                                ;{background-color: {self.parent.standard_hover_color
                                {{
                                }} QPushButton
                                ;background-color: transparent
                                {{
                                }} QPushButton: pressed
                                ;background-color: #202225
                                ;border-color: #72767d
                                (""""{{

```

```
((button.clicked.connect(lambda checked: click_callback(friend
                                                                    (button.move(x, y
```

```
                                                                    (label.setText(label_text
                                                                    ()label.hide
(";label.setStyleSheet("color: white; font-size: 12px
                                                                    (label.move(x, y - 20
```

```
Connect the label to show on hover #
()button.enterEvent = lambda event: label.show
()button.leaveEvent = lambda event: label.hide
```

```
:(def on_text_changed_in_contact_search(self
This function will be called when the text inside QLineEdit changes #
                                                                    [] = default_list
:"if self.parent.friends_box_page == "online
default_list = self.parent.online_users_list
:"elif self.parent.friends_box_page == "all
default_list = self.parent.friends_list
:"elif self.parent.friends_box_page == "pending
default_list = self.parent.friends_list
:"elif self.parent.friends_box_page == "blocked
default_list = self.parent.friends_list
                                                                    :try
                                                                    :()if self.search.hasFocus
                                                                    :if len(self.search.text()) > 0
self.parent.current_friends_box_search = True
```

```

        self.parent.temp_search_list =
(filter_and_sort_chats(self.search.text(), default_list
        )self.parent.updated_social_page
                                :else
                                :try
self.parent.current_friends_box_search = False
        [] = self.parent.temp_search_list
        ()self.parent.updated_social_page
                                :except Exception as e
        ("{print(f"problem with updating screen:{e
                                :except Exception as e
        ("{print(f"problem with updating screen:{e

```

```

                                : (def raise_all_element(self
                                ()_self.border1_label.raise
                                ()_self.border2_label.raise
                                ()_self.add_friend.raise
                                ()_self.all_button.raise
                                ()_self.Pending_button.raise
                                ()_self.online_button.raise
                                ()_self.friends_label.raise
                                ()_self.blocked_button.raise
                                ()_self.search.raise
                                ()_self.social_label.raise
                                :try
                                ()_self.block_friend_label.raise
                                ()_self.remove_friend_label.raise
                                ()_self.chat_label.raise
                                :except Exception as e

```

```

x = 4

:for element in self.raised_elements
    ()_element.raise

:(def is_mouse_on_friends_box(self, mouse_pos
    ()box_geometry = self.border3_label.geometry
    (return box_geometry.contains(mouse_pos

:(def online_button_pressed(self
    :if self.parent.friends_box_page != "online
    :if self.parent.friends_box_page != "add friend
        ()self.search.clear
self.parent.current_friends_box_search = False
    [] = self.parent.temp_search_list
    self.parent.friends_box_index = 0
    "self.parent.friends_box_page = "online
    ()self.parent.updated_social_page

:(def all_button_pressed(self
    :if self.parent.friends_box_page != "all
    :if self.parent.friends_box_page != "add friend
        ()self.search.clear
self.parent.current_friends_box_search = False
    [] = self.parent.temp_search_list
    self.parent.friends_box_index = 0
    "self.parent.friends_box_page = "all
    ()self.parent.updated_social_page

```

```

:(def pending_button_pressed(self
                                :try
                                :if self.parent.friends_box_page != "pending
:"if self.parent.friends_box_page != "add friend
                                ()self.search.clear
self.parent.current_friends_box_search = False
                                [] = self.parent.temp_search_list
                                self.parent.friends_box_index = 0
                                "self.parent.friends_box_page = "pending
                                ()self.parent.updated_social_page
                                :except Exception as e
                                ("{print(f"error pending_button_pressed {e

```

```

:(def blocked_button_pressed(self
                                :if self.parent.friends_box_page != "blocked
:"if self.parent.friends_box_page != "add friend
                                ()self.search.clear
self.parent.current_friends_box_search = False
                                [] = self.parent.temp_search_list
                                self.parent.friends_box_index = 0
                                "self.parent.friends_box_page = "blocked
                                ()self.parent.updated_social_page

```

```

:(def add_friend_button_pressed(self
                                :try
                                :if self.parent.friends_box_page != "add friend
:"if self.parent.friends_box_page != "add friend
                                ()self.search.clear

```

```

self.parent.current_friends_box_search = False
        [] = self.parent.temp_search_list
        self.parent.friends_box_index = 0
        "self.parent.friends_box_page = "add friend
        ()self.parent.updated_social_page
        :except Exception as e
        ("{{print(f"error add_friend_button_pressed {e

        : (def open_chat(self, friend
Implement the logic to start a chat with the selected friend #
        ("{{print(f"Starting chat with {friend
        (self.parent.chat_box.selected_chat_changed(friend
        ()self.parent.chat_clicked

        : (def remove_friend(self, friend
Implement the logic to start a chat with the selected friend #
        ("print(f"Removing {friend} as friend
        (self.Network.send_remove_friend(friend
        (self.parent.friends_list.remove(friend
        ()self.parent.updated_social_page

        : (def block_friend(self, friend
Implement the logic to start a chat with the selected friend #
        (self.Network.block_user(friend
        ("{{print(f"blocking {friend

        : (def unblock_friend(self, friend
Implement the logic to start a chat with the selected friend #

```



```

        (self.Network.unblock_user(friend
            ("){print(f'unblocked {friend

:(def draw_circle(self, widget, color_of_circle
    (pixmap = QPixmap(20, 20
    ((pixmap.fill(QColor(color_of_circle

        (widget.setPixmap(pixmap

:(def send_friend_request(self
Implement the logic to send friend requests here #
        ()self.error_friend.hide
        ()self.request_sent.hide
        ()self.already_friends.hide
        ()self.already_sent_request.hide
:if self.add_friend_entry.hasFocus() and len(self.add_friend_entry.text()) > 0
    username = self.username
    ()friend_username = self.add_friend_entry.text
    (self.Network.send_friend_request(friend_username
        ("")self.add_friend_entry.setText

:(def friend_not_found(self
    ()self.error_friend.show

:(def request_was_sent(self
    ()self.request_sent.show

:(def request_was_friend(self

```

```

        ()self.already_friends.show

        :(def request_is_pending(self
        ()self.already_sent_request.show

        :(def handle_friend_request(self, index, accept=True
        Handle friend requests (accept or reject) here #
            friend_index = index // 3
            [friend_label = self.requests_items[index
            ()friend_username = friend_label.text

            :if accept
        (self.Network.send_friends_request_acception(friend_username
            (self.parent.request_list.remove(friend_username
            :else
            (self.parent.request_list.remove(friend_username
        (self.Network.send_friends_request_rejection(friend_username

        :(for i in range(index, index + 3
        [item = self.requests_items[i
            (item.setParent(None
            ()item.deleteLater

        Remove the friend from the requests list #
        [del self.requests_items[index:index + 3
        ()self.parent.updated_social_page

        * from PyQt5.QtWidgets import

```

```

from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit
from PyQt5.QtCore import QSize, QTimer, Qt, QUrl, QTime, QTemporaryFile
from PyQt5.QtGui import QIcon, QPixmap
from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent
from PyQt5.QtMultimediaWidgets import QVideoWidget
from io import BytesIO
from PIL import Image, ImageDraw
import pyaudio
import cv2
from datetime import datetime
from settings_page_widgets import create_slider
from messages_page_widgets import make_q_object_clear, set_button_icon,
set_icon_from_path_to_label, open_image_bytes

```

```

:(def insert_item_to_table(table, col, value, row_position
    if col == 3: # If it's the column for the photo
        ()item = QTableWidgetItem
        ()pixmap = QPixmap
        (pixmap.loadFromData(value
        ((item.setIcon(QIcon(pixmap
        ((item.setSizeHint(QSize(100, 100
        :else
        ((item = QTableWidgetItem(str(value
        :if col != 0
        (item.setTextAlignment(Qt.AlignCenter
        (table.setItem(row_position, col, item

```

```

        :()def remove_row(table, row_number
:()if row_number >= 0 and row_number < table.rowCount
        (table.removeRow(row_number
        ("print(f"Row {row_number} removed successfully
        :else
        ("print("Invalid row number. Row not removed

:()def get_camera_names
        [] = camera_names
for i in range(10): # You can adjust the range according to your needs
        (cap = cv2.VideoCapture(i
        :()if cap.isOpened
        ("camera_names.append(f"Camera {i
        ()cap.release
        :else
        break
        return camera_names

:()def get_default_output_device_name
        ()p = pyaudio.PyAudio
()return p.get_default_output_device_info().get("name").lower

:()def get_default_input_device_name
        ()p = pyaudio.PyAudio
()return p.get_default_input_device_info().get("name").lower

```

```

:(def find_input_device_index(device_name
                                ()p = pyaudio.PyAudio
                                input_device_index = None

                                :())for i in range(p.get_device_count
                                (device_info = p.get_device_info_by_index(i
                                :())if device_info["name"].lower() == device_name.lower
                                input_device_index = i
                                break

                                ()p.terminate
                                return input_device_index

```

```

:(def find_output_device_index(device_name
                                ()p = pyaudio.PyAudio
                                output_device_index = None

                                :())for i in range(p.get_device_count
                                (device_info = p.get_device_info_by_index(i
                                :())if device_info["name"].lower() == device_name.lower
                                output_device_index = i
                                break

                                ()p.terminate
                                return output_device_index

```

```

:(def try_to_open_output_stream(index
    audio_format = pyaudio.paInt16
        channels = 1
        rate = 44100
        chunk = 1024
    )p = pyaudio.PyAudio
        :try
output_stream = p.open(format=audio_format, channels=channels,
    ,rate=rate, output=True, frames_per_buffer=chunk
    (output_device_index=index
        return True
        :except
    return False

```

```

:(def try_to_open_input_stream(index
    audio_format = pyaudio.paInt16
        channels = 1
        rate = 44100
        chunk = 1024
    )p = pyaudio.PyAudio
        :try
,input_stream = p.open(format=audio_format
    ,channels=channels
        ,rate=rate
        ,input=True
    ,frames_per_buffer=chunk

```

```

        (input_device_index=index
            return True
        ):except
        return False

:()def get_output_devices
    [] = input_devices
    ()p = pyaudio.PyAudio
    :()for i in range(p.get_device_count
        (device_info = p.get_device_info_by_index(i
            ()device_name = device_info["name"].lower
            if device_info["maxOutputChannels"] > 0 and ("headset" in device_name or
                :("speaker" in device_name or "headphones" in device_name
                    :if device_info["name"] not in input_devices
                        device_index = i
                    :(if try_to_open_output_stream(device_index
                        (["input_devices.append(device_info["name
                            ()p.terminate
                        return input_devices

:()def get_input_devices
    [] = output_devices
    ()p = pyaudio.PyAudio
    :()for i in range(p.get_device_count
        (device_info = p.get_device_info_by_index(i
            ()device_name = device_info["name"].lower
            :(if device_info["maxInputChannels"] > 0 and ("microphone" in device_name

```

```

        :if device_info["name"] not in output_devices
            device_index = i
        :if try_to_open_input_stream(device_index
(["output_devices.append(device_info["name
                                ()p.terminate
                                return output_devices

```

```

:(def format_label_text_by_row(label, text, num_rows
                                :try

```

Calculate the total number of characters per row, including the possibility #
of a shorter last row

```

chars_per_row = len(text) // num_rows
extra_chars = len(text) % num_rows

```

Initialize variables for storing formatted text and the starting index #

```

"" = formatted_text
start_index = 0

```

Iterate through each row #

```

:(for row in range(num_rows

```

Calculate the end index for this row #

```

end_index = start_index + chars_per_row

```

Add an extra character to this row if needed #

```

:if row < extra_chars
    end_index += 1

```

Add the substring to the formatted text with a newline character #


```
"formatted_text += text[start_index:end_index] + "\n
```

```
Update the starting index for the next row #
```

```
start_index = end_index
```

```
Set the formatted text to the label #
```

```
(label.setText(formatted_text
```

```
:except Exception as e
```

```
("{print(f" error in creating formatted label by rows: {e
```

```
:(def make_circular_image(image_bytes
```

```
.Converts an image to a circular image with the same width and height""
```

```
:Args
```

```
.image_bytes (bytes): Image data as bytes
```

```
:Returns
```

```
.bytes: The circular image as bytes
```

```
""
```

```
:try
```

```
Load the image using Pillow #
```

```
:with BytesIO(image_bytes) as image_buffer
```

```
(image = Image.open(image_buffer
```

```
(Convert the image to RGBA mode (if not already #
```

```
:if image.mode != 'RGBA
```

```
('image = image.convert('RGBA
```

```

Determine the minimum dimension for the circular image #
(min_dimension = min(image.width, image.height

Create a square-shaped image #
square_image = Image.new('RGBA', (min_dimension, min_dimension),
                           (((255, 255, 255, 0
offset = ((min_dimension - image.width) // 2, (min_dimension -
                                                (image.height) // 2
(square_image.paste(image, offset

Determine the maximum circular area based on the minimum #
dimension
max_radius = min_dimension // 2

Create a mask with a transparent circle #
mask = Image.new('L', (min_dimension, min_dimension), 0) # Create a
black mask
(draw = ImageDraw.Draw(mask
(draw.ellipse(((0, 0), (min_dimension, min_dimension))), fill=255

Apply the mask to the image #
circular_image = Image.new('RGBA', (min_dimension, min_dimension),
                           (((255, 255, 255, 0
(circular_image.paste(square_image, mask=mask

Convert the circular image to bytes #
()output_buffer = BytesIO
circular_image.save(output_buffer, format='PNG') # Save as PNG format
()circular_image_bytes = output_buffer.getvalue

```

```

        return circular_image_bytes

    except Exception as e:
        print(f"Error converting image: {e}")
        return None

class VideoPlayer(QWidget):
    def __init__(self, video_bytes, parent=None):
        super().__init__(parent)
        self.parent = parent
        self.video_bytes = video_bytes
        self.setWindowTitle("Video Player")

    def __init__(self, video_bytes, parent=None):
        self.media_player = QMediaPlayer(None, QMediaPlayer.VideoSurface)
        self.video_widget = QVideoWidget(self)
        screen = QDesktopWidget().screenGeometry()
        # Extract the screen width and height
        screen_width = screen.width()
        screen_height = screen.height()
        self.video_widget.setGeometry(0, 0, screen_width, int(screen_height*0.83))

        self.slider = QSlider(Qt.Horizontal, self)
        self.slider.sliderReleased.connect(self.set_position)
        self.slider_x, self.slider_y, self.slider_width, self.slider_height = int(screen_width*0.1),
            int(screen_height*0.85), int(screen_width*0.8), int(screen_height*0.05)
        self.slider.setGeometry(self.slider_x, self.slider_y, self.slider_width, self.slider_height)

```

```

        (exit_watch_button = QPushButton(self
exit_watch_button_x, exit_watch_button_y = (slider_x + slider_width + 20,
        ((slider_y + int(screen_height * 0.005
        (make_q_object_clear(exit_watch_button
        (exit_watch_button.clicked.connect(self.stop_watching

        "icon_path = "discord_app_assets/exit_button.png
        (button_size = (30, 30
        set_button_icon(exit_watch_button, icon_path, button_size[0],
        ([button_size[1
exit_watch_button.setGeometry(exit_watch_button_x, exit_watch_button_y,
        ([button_size[0], button_size[1

        (self.duration_label = QLabel(self
self.duration_label.setStyleSheet("background-color: transparent; color:
        (";white
        duration_label_x, duration_label_y = int(screen_width*0.03),
        (int(screen_height*0.865
        (self.duration_label.move(duration_label_x, duration_label_y

self.video_widget.mousePressEvent = self.toggle_play_pause

        Set media player to use video widget #
        (self.media_player.setVideoOutput(self.video_widget
        (self.media_player.durationChanged.connect(self.update_duration
        (self.media_player.positionChanged.connect(self.update_position
        (self.media_player.stateChanged.connect(self.handle_state_change
        (self.media_player.setVolume(self.parent.volume

        (self.position_timer = QTimer(self

```

```

(self.position_timer.timeout.connect(self.update_slider_position

(Start the position update timer with a short interval (e.g., 1 milliseconds #
    (self.position_timer.start(1

                                :(def update_slider_position(self
Update the slider position based on the current media player position #
    ()position = self.media_player.position
    (self.slider.setValue(position
        Format the position and duration to MM:SS format #
    ("position_time = QTime(0, 0).addMSecs(position).toString("mm:ss
        duration_time = QTime(0,
        ("0).addMSecs(self.media_player.duration()).toString("mm:ss
        Update the duration label text #
    ("{self.duration_label.setText(f"{position_time} / {duration_time
    (self.slider.setValue(position

                                :(def toggle_play_pause(self
:if self.media_player.state() == QMediaPlayer.PlayingState
    ()self.media_player.pause
    ()self.position_timer.stop
    :else
    ()self.media_player.play
    ()self.position_timer.start

                                :(def stop_watching(self
    ()self.media_player.stop
    ()self.parent.stop_watching_video

```

```

                                :(def play_video(self
                                    :try
                                Create a temporary file in memory #
                                    ()temp_file = QTemporaryFile
                                    :()if temp_file.open
                                Write the video bytes to the temporary file #
                                    (temp_file.write(self.video_bytes
                                Flush the data to ensure it's written #
                                    ()temp_file.flush

                                Create a QMediaContent object with the QUrl pointing to the #
                                    temporary file
                                (()url = QUrl.fromLocalFile(temp_file.fileName
                                    (media_content = QMediaContent(url

                                Set the media content to the media player and play #
                                    (self.media_player.setMedia(media_content
                                    ()self.media_player.play
                                    :else
                                ("print("Failed to create temporary file
                                    :except Exception as e
                                ("{print(f"play_video error :{e

                                :(def update_duration(self, duration
                                (self.slider.setMaximum(duration
                                ()self.duration_label.adjustSize
                                ()_self.duration_label.raise

                                :(def update_position(self, position

```

```

        (self.slider.setValue(position
        Format the position and duration to MM:SS format #
        ("position_time = QTime(0, 0).addMSecs(position).toString("mm:ss
        duration_time = QTime(0,
        ("0).addMSecs(self.media_player.duration()).toString("mm:ss
        Update the duration label text #
        ("{self.duration_label.setText(f"{position_time} / {duration_time
        if position >= self.media_player.duration() - 10: # Check if less than 100
        milliseconds remain
        ()self.media_player.pause
        self.media_player.setPosition(0) # Rewind to the beginning for replay

        :(def handle_state_change(self, new_state
        :if new_state == QMediaPlayer.EndOfMedia
        If video reaches the end, pause instead of finishing #
        ()self.play_video
        ()self.media_player.pause

        :(def set_position(self
        ()position = self.slider.value
        (self.media_player.setPosition(position

        :(def keyPressEvent(self, event
        :if event.key() == Qt.Key_Space
        :if self.media_player.state() == QMediaPlayer.PlayingState
        ()self.media_player.pause
        ()self.position_timer.stop
        :elif self.media_player.state() == QMediaPlayer.PausedState
        ()self.media_player.play

```

```

        ()self.position_timer.start
    elif event.key() == Qt.Key_Escape
        ()self.stop_watching

class PlaylistWidget(QWidget):
    def __init__(self, main_page_widget):
        super().__init__()
        self.parent = main_page_widget
        self.init_ui()

    def init_ui(self):
        # Create a table widget #
        try:
            self.sliders_style_sheet = f"""
            QSlider::groove:horizontal
            {
                border: 1px solid #bbb
                background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0
                {
                    ;(#ddd, stop:1 #eee
                }
                ;height: 10px
                ;margin: 0px
            }

            QSlider::handle:horizontal
            {
                background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0
                {
                    ;(#eee, stop:1 #ccc
                }
                ;border: 1px solid #777
                ;width: 20px
            }

```



```

margin: -2px 0; /* handle is placed by default on the contents
                /* rect of the groove. Expand outside the groove

                ;border-radius: 5px

                {{

                }} QSlider::add-page:horizontal
                ;background: #fff

                {{

                }} QSlider::sub-page:horizontal
background: {self.parent.standard_hover_color}; /* Change
                /* this color to the desired color for the left side

                {{

                """"

                self.last_selected_row = None

                table_x, table_y = 0, 70

                table_width, table_height = int(self.parent.screen_width * 0.99),
                (int(self.parent.screen_height * 0.075

self.search_table = self.create_table_widget(table_width, table_height,
                ("", table_x, table_y

                (self.search_table.insertRow(0

                ()self.search_table.clearSelection

                (self.search_table.setSelectionMode(QAbstractItemView.NoSelection

                table_x, table_y = 0, self.parent.screen_height // 5.4

                table_width, table_height = int(self.parent.screen_width * 0.99),
                (int(self.parent.screen_height * 0.648

self.table = self.create_table_widget(table_width, table_height, table_x,
                ("table_y, "playlist_table

```

```

        (self.search_song_entry = QLineEdit(self
        search_song_entry_x, search_song_entry_y =
            int(self.parent.screen_width * 0.35), 0
            width, height = 450, 40
        self.search_song_entry.setGeometry(search_song_entry_x,
            (search_song_entry_y, width, height
        self.search_song_entry.setPlaceholderText("🔍 What do you want to
            ("?play

        self.add_to_playlist_button = QPushButton("➕ Add song to Playlist",
            (self
            (make_q_object_clear(self.add_to_playlist_button
            size = QSize(int(0.072 * self.parent.screen_width),
                ((int(self.parent.screen_height * 0.028
                (self.add_to_playlist_button.setFixedSize(size
            add_to_playlist_button_x, add_to_playlist_button_y = 0,
                (int(self.parent.screen_height * 0.03
            self.add_to_playlist_button.move(add_to_playlist_button_x,
                (add_to_playlist_button_y

        self.add_to_playlist_button.clicked.connect(self.parent.save_searched_song_to_
            (playlist

        self.try_searched_song_button = QPushButton("🎧 Check out the song",
            (self
            (make_q_object_clear(self.try_searched_song_button
            (self.try_searched_song_button.setFixedSize(size
            try_searched_song_button_x, try_searched_song_button_y =
                )int(size.width() * 1.1), int
                (self.parent.screen_height * 0.03
            self.try_searched_song_button.move(try_searched_song_button_x,
                (try_searched_song_button_y

```

```

(self.try_searched_song_button.clicked.connect(self.parent.play_search_result

self.remove_selected_song_button = QPushButton("✖ Remove selected
                                                (song", self

        (make_q_object_clear(self.remove_selected_song_button
            (self.remove_selected_song_button.setFixedSize(size
remove_selected_song_button_x, remove_selected_song_button_y = 0,
                                                                    )int
                                                                    (self.parent.screen_height * 0.15

self.remove_selected_song_button.move(remove_selected_song_button_x,
                                        (remove_selected_song_button_y

(self.remove_selected_song_button.clicked.connect(self.remove_song
        (self.playlist_duration_slider_current_time_label = QLabel(self
            (self.playlist_duration_slider_duration_label = QLabel(self
(make_q_object_clear(self.playlist_duration_slider_current_time_label
        (make_q_object_clear(self.playlist_duration_slider_duration_label
            playlist_duration_slide_x, playlist_duration_slide_y =
                )int(self.parent.screen_width * 0.265), int
                (self.parent.screen_height * 0.15
            playlist_slider_width, playlist_slider_height = 750, 25

self.playlist_duration_slider = create_slider(self, 0, 0, 0, None
        playlist_duration_slide_x ,
        playlist_duration_slide_y, ,
        (playlist_slider_width, playlist_slider_height, self.sliders_style_sheet

self.playlist_duration_slider.sliderMoved.connect(self.update_media_player_posit
                                                                    (ion

```

```

self.playlist_duration_slider_duration_label.move(int((playlist_duration_slide_x +
                                                    ,(playlist_slider_width)*1.01

                                                    ((int(playlist_duration_slide_y*1.035

self.playlist_duration_slider_current_time_label.move(int(playlist_duration_slide_
                                                    ,(x*0.92

                                                    ((int(playlist_duration_slide_y*1.035
self.playlist_duration_slider_current_time_label.setStyleSheet("color:
                                                    ("white

("self.playlist_duration_slider_duration_label.setStyleSheet("color: white

                                                    (last_song_button = QPushButton(self
                                                    (next_song_button = QPushButton(self
                                                    (pause_and_play_button = QPushButton(self
                                                    (self.shuffle_button = QPushButton(self
                                                    (self.replay_song_button = QPushButton(self
"last_song_button_icon_path = "discord_app_assets/last_song_icon.png
                                                    next_song_button_icon_path =
                                                    ""discord_app_assets/next_song_icon.png
                                                    pause_and_play_button_icon_path =
                                                    ""discord_app_assets/pause_and_play_icon.png
"shuffle_button_icon_path = "discord_app_assets/suffle_icon.png
                                                    replay_song_button_icon_path =
                                                    ""discord_app_assets/replaying_icon.png
buttons_width, buttons_height = int(self.parent.screen_width*0.026),
                                                    (int(self.parent.screen_height*0.0462
set_button_icon(self.replay_song_button, replay_song_button_icon_path,
                                                    (buttons_width, buttons_height
set_button_icon(last_song_button, last_song_button_icon_path,
                                                    (buttons_width, buttons_height

```

```

set_button_icon(next_song_button, next_song_button_icon_path,
                (buttons_width, buttons_height)

                set_button_icon(pause_and_play_button,
                (pause_and_play_button_icon_path, buttons_width, buttons_height)

                set_button_icon(self.shuffle_button, shuffle_button_icon_path,
                (buttons_width, buttons_height)

                (first_button_x = int(self.parent.screen_width * 0.43
                (buttons_y = int(self.parent.screen_height * 0.842
playlist_volume_slider_x, playlist_volume_slider_y = int(buttons_width *
                )1.2), int
                (self.parent.screen_height * 0.854
playlist_volume_slider_width, playlist_volume_slider_height = 500, 25
                (self.playlist_volume_slider_label = QLabel(self
                (make_q_object_clear(self.playlist_volume_slider_label
self.playlist_volume_slider = create_slider(self, 0, 100, self.parent.volume
                ,self.playlist_volume_update ,
                playlist_volume_slider_x,
                playlist_volume_slider_y, playlist_volume_slider_width,
                playlist_volume_slider_height, self.sliders_style_sheet
                (
self.playlist_volume_slider_label.move(int(playlist_volume_slider_x*0.4),
                ((int(playlist_volume_slider_y*0.995
                ()self.set_icon_for_volume_label
                .self.volume_slider #

                (delta_between_button = int(self.parent.screen_width * 0.03125
self.replay_song_button.move(first_button_x+(delta_between_button*3),
                (buttons_y
(self.shuffle_button.move(first_button_x-delta_between_button, buttons_y
                (last_song_button.move(first_button_x, buttons_y

```

```

pause_and_play_button.move(first_button_x+delta_between_button,
                             (buttons_y
next_song_button.move(first_button_x+(delta_between_button*2),
                             (buttons_y

        (make_q_object_clear(last_song_button
        (make_q_object_clear(next_song_button
        (make_q_object_clear(pause_and_play_button
        (make_q_object_clear(self.shuffle_button
        (make_q_object_clear(self.replay_song_button
                                ()_last_song_button.raise
                                ()_next_song_button.raise
                                ()_pause_and_play_button.raise

pause_and_play_button.clicked.connect(self.parent.pause_and_unpause_playlis
(t

        (last_song_button.clicked.connect(self.parent.go_to_last_song
        (next_song_button.clicked.connect(self.parent.go_to_next_song
        (self.shuffle_button.clicked.connect(self.toggle_shuffle
        (self.replay_song_button.clicked.connect(self.toggle_replay_song

                                ()self.update_music_page_style_sheet
                                Ensure the data is visible #

                                ()self.table.show
                                :except Exception as e
                                ("{print(f"error in creating music box {e

:(def update_media_player_position(self, new_position
    Convert the new position to seconds #

                                :try

```

```

:if self.parent.playlist_media_player.state() == QMediaPlayer.PlayingState
    is_playing = True
:else
    is_playing = False
    ()self.parent.playlist_media_player.pause
new_position_seconds = new_position # Convert milliseconds to
                                   seconds

```

```

    Set the new position of the media player #
(self.parent.playlist_media_player.setPosition(new_position_seconds
    :if is_playing
    ()self.parent.playlist_media_player.play
    :except Exception as e
    ("{print(f"error with changing audio position {e

```

```

:(def update_current_duration_text(self, text
(self.playlist_duration_slider_current_time_label.setText(text
()self.playlist_duration_slider_current_time_label.adjustSize

```

```

:(def update_duration_tex(self, text
(self.playlist_duration_slider_duration_label.setText(text
()self.playlist_duration_slider_duration_label.adjustSize

```

```

:(def set_icon_for_volume_label(self
"muted_volume_path = "discord_app_assets/speaker_icon0.png
"low_volume_path = "discord_app_assets/speaker_icon1.png
"mid_volume_path = "discord_app_assets/speaker_icon2.png
"high_volume_path = "discord_app_assets/speaker_icon3.png
:if self.parent.playlist_volume == 0

```

```

set_icon_from_path_to_label(self.playlist_volume_slider_label,
                             (muted_volume_path
                              :elif self.parent.playlist_volume < 30
set_icon_from_path_to_label(self.playlist_volume_slider_label,
                             (low_volume_path
                              :elif self.parent.playlist_volume < 60
set_icon_from_path_to_label(self.playlist_volume_slider_label,
                             (mid_volume_path
                              :else
set_icon_from_path_to_label(self.playlist_volume_slider_label,
                             (high_volume_path

:(def playlist_volume_update(self, value
  (self.playlist_volume_slider.setValue(value
    self.parent.playlist_volume = value
  (self.parent.playlist_media_player.setVolume(value
    ()self.set_icon_for_volume_label

:(def toggle_shuffle(self
  ()self.parent.toggle_shuffle
  :if self.parent.shuffle
self.shuffle_button.setStyleSheet(f"background-color:
  "{{self.parent.standard_hover_color
  (";f"; border-radius: 15px
  :else
"self.shuffle_button.setStyleSheet(f"background-color: transparent
  (";f"; border-radius: 15px

:(def toggle_replay_song(self
  :try

```



```

(table.setGeometry(table_x, table_y, table_width, table_height

                                : "if table_name == "playlist_table
                                (table.cellPressed.connect(self.cell_pressed
(table.itemSelectionChanged.connect(self.onSelectionChanged

                                (table.setShowGrid(False
                                first_column_width = table_width * 0.4
                                other_column_width = table_width * 0.18

                                (table.setColumnWidth(0, first_column_width
                                (table.setColumnWidth(1, other_column_width
                                (table.setColumnWidth(2, other_column_width
                                (table.setColumnWidth(3, other_column_width

                                return table

                                : (def update_music_page_style_sheet(self
                                (self.apply_table_stylesheet(self.table
                                (self.apply_table_stylesheet(self.search_table
                                ()self.apply_style_sheet_to_text_entry
                                ()self.apply_style_sheet_for_button

                                : (def apply_style_sheet_to_text_entry(self
                                : "if self.parent.background_color == "Black and White
                                "text_entry_color = "black
                                :else

```

```

        "text_entry_color = "white
    )self.search_song_entry.setStyleSheet
" ;{f"background-color: {self.parent.standard_hover_color
    " ;f"color: {text_entry_color}; padding: 10px
    (" ;f"border-radius: 5px; font-size: 14px

:(def apply_style_sheet_for_button(self
:"if self.parent.background_color == "Black and White
    "text_entry_color = "black
    :else
    "text_entry_color = "white
style_sheet = f"background-color: {self.parent.standard_hover_color}; color:
    "{text_entry_color}; border-radius: 15px
    (self.add_to_playlist_button.setStyleSheet(style_sheet
    (self.try_searched_song_button.setStyleSheet(style_sheet
    (self.remove_selected_song_button.setStyleSheet(style_sheet

:(def apply_table_stylesheet(self, table
:"if self.parent.background_color == "Black and White
    "text_entry_color = "black
    :else
    "text_entry_color = "white
    ""table.setStyleSheet(f
        }} QTableWidgetItem
    ;{background-color: {self.parent.background_color_hex
        ;border: 1px solid #d0d0d0
    ;{selection-background-color: {self.parent.standard_hover_color
        ;{color: {text_entry_color
        {{

```

```

        }} QTableWidgetItem::item:hover
        ;{background-color: {self.parent.standard_hover_color
        {{
        }} QTableWidgetItem QHeaderView::section
        ;{color: {text_entry_color
/* border: 0; /* Remove border between column headers
        {{
        }} QHeaderView::section
        ;{color: {text_entry_color
        ;{background-color: {self.parent.background_color_hex
        ;border: 1px solid #d0d0d0
        ;padding: 4px
        {{
        }} QHeaderView::section:checked
        ;{background-color: {self.parent.standard_hover_color
        {{
        }} QTableWidgetItem:selected
        ;{background-color: {self.parent.standard_hover_color
        {{
        }} QTableWidgetItem QTableWidgetItem
        ;{color: {text_entry_color
/* text-align: center; /* Align text in the middle
/* border: 0; /* Remove border between cells
        {{
        (""

        : (def find_row_by_text(self, text
        : ((for row in range(self.table.rowCount

```

```

(item = self.table.item(row, 0) # Assuming the first column (index 0
if item and (item.text() == text or item.text().startswith(text) or text in
                                                    :((item.text
                                                    return row
return -1 # Return -1 if the text is not found in any row

```

```

:(def insert_search_data(self, video_info_dict
                                :try
                                row_position = 0
                                Extract data from the dictionary #
                                ('title = video_info_dict.get('title
('thumbnail_bytes = video_info_dict.get('thumbnail_bytes
('audio_bytes = video_info_dict.get('audio_bytes
('audio_duration = video_info_dict.get('audio_duration

                                "Set the current date as the "Date Added #
                                ('date_added = datetime.now().strftime('%Y-%m-%d

                                Insert data into the table #
                                for col, value in enumerate([title, date_added, audio_duration,
                                                                :([thumbnail_bytes
(insert_item_to_table(self.search_table, col, value, row_position
                                                                :except Exception as e
                                                                (print(e

```

```

:(def insert_playlist_songs(self, list_video_info_dict
                                :try
                                row_position = 0
                                :for video_info_dict in list_video_info_dict

```

```

        (self.table.insertRow(row_position
        Extract data from the dictionary #
        ('title = video_info_dict.get('title
('thumbnail_bytes = video_info_dict.get('thumbnail_bytes
        ('audio_duration = video_info_dict.get('audio_duration
        ('date_added = video_info_dict.get('timestamp

        Insert data into the table #
for col, value in enumerate([title, date_added, audio_duration,
                             :([thumbnail_bytes
(insert_item_to_table(self.table, col, value, row_position
                             row_position += 1
                             (self.select_row(0
                             :except Exception as e
                             (print(e

:(def insert_new_song_to_playlist(self, song_dict
                             :try

        ()row_position = self.table.rowCount
        (self.table.insertRow(row_position
        ('title = song_dict.get('title
('thumbnail_bytes = song_dict.get('thumbnail_bytes
        ('audio_duration = song_dict.get('audio_duration
('date_added = datetime.now().strftime('%Y-%m-%d

        Insert data into the table #
for col, value in enumerate([title, date_added, audio_duration,
                             :([thumbnail_bytes
(insert_item_to_table(self.table, col, value, row_position

```

```

        :except Exception as e
        ({print(f"error in insert_new_song_to_playlist {e

        :(def cell_pressed(self, row, col
        Get the item text when a cell is pressed #
        :if col < 3
        (self.parent.set_new_playlist_index_and_listen(row
        (self.select_row(row
        (item = self.table.item(row, col
        :if item
        (())print("Cell Pressed:", item.text
        :if col == 3
        means album image pressed #
        [song_dict = self.parent.playlist_songs[row
        ("thumbnail_bytes = song_dict.get("thumbnail_bytes
        (open_image_bytes(thumbnail_bytes

        :(def select_row(self, row
        Clear any existing selections #
        :try
        self.last_selected_row = row
        ()self.table.clearSelection
        Create a selection range for the entire row #
        selection_range = QTableWidgetItemSelectionRange(row, 0, row,
        (self.table.columnCount() - 1

        Select the range #
        (self.table.setRangeSelected(selection_range, True
        ()self.table.horizontalHeader().clearSelection#

```

```

        :except Exception as e
        ("{print(f"error in inserting table {e

        :(def clear_selection(self
        ()self.table.clearSelection

        :(def onSelectionChanged(self
        :try
        :if self.last_selected_row is not None
        ()selected_rows = self.table.selectionModel().selectedRows
        :if len(selected_rows) == 0
        No rows are selected, so reselect the previously selected row #
        (self.select_row(self.last_selected_row
        :except Exception as e
        (print(e

* from PyQt5.QtWidgets import
from PyQt5.QtWidgets import QWidget, QLabel
from PyQt5.QtCore import QSize, pyqtSignal, Qt
from PyQt5.QtGui import QIcon, QPixmap
from io import BytesIO
from PIL import Image, ImageDraw
import re
import pyaudio
import cv2

```



```

:(def handle_combobox_visibility_changed(is_visible, arrow_label
                                         :if is_visible
                                         )arrow_label.setPixmap
QPixmap("discord_app_assets/up_arrow_icon.png").scaledToWidth(30,
                      ((Qt.SmoothTransformation
                      :else
                      )arrow_label.setPixmap

QPixmap("discord_app_assets/down_arrow_icon.png").scaledToWidth(30,
                      ((Qt.SmoothTransformation

:(def is_valid_image(image_bytes
                     :try
                     Use Pillow to try opening the image from bytes #
                     ((image = Image.open(BytesIO(image_bytes
                     If successful, it's a valid image #
                     return True
                     :except Exception as e
                     If there is an exception, it's not a valid image #
                     ("{print(f"Error: {e
                     return False

def create_slider(parent, min_value, max_value, value, connected_function, x, y,
                  :(width, height, style_sheet

(volume_slider = QSlider(Qt.Horizontal, parent
                        (volume_slider.setMinimum(min_value
                        (volume_slider.setMaximum(max_value
volume_slider.setValue(value) # Set initial volume

```

```

        :if connected_function is not None
(volume_slider.valueChanged.connect(connected_function
    (volume_slider.setGeometry(x, y, width, height
        (volume_slider.setStyleSheet(style_sheet
            return volume_slider

:(def file_to_bytes(file_path
:with open(file_path, "rb") as file
    ()image_bytes = file.read
        return image_bytes

:(def create_custom_circular_label(width, height, parent
    (label = QLabel(parent

    (label_size = QSize(width, height
        (label.setFixedSize(label_size

        """))label.setStyleSheet
            } QLabel
border-radius: "" + str(height // 2) + ""px; /* Set to half of the label height
/*
background-color: transparent; /* Make the background color transparent
/*

{
    (""

```

```
return label
```

```
:(def try_to_open_output_stream(index
    audio_format = pyaudio.paInt16
        channels = 1
        rate = 44100
        chunk = 1024
    )p = pyaudio.PyAudio
        :try
output_stream = p.open(format=audio_format, channels=channels,
    ,rate=rate, output=True, frames_per_buffer=chunk
    (output_device_index=index
        return True
        :except
return False
```

```
:(def try_to_open_input_stream(index
    audio_format = pyaudio.paInt16
        channels = 1
        rate = 44100
        chunk = 1024
    )p = pyaudio.PyAudio
        :try
,input_stream = p.open(format=audio_format
    ,channels=channels
        ,rate=rate
```

```

        ,input=True
        ,frames_per_buffer=chunk
        (input_device_index=index
            return True
        ):except
        return False

:()def get_output_devices
    [] = input_devices
    ()p = pyaudio.PyAudio
    :()for i in range(p.get_device_count
        (device_info = p.get_device_info_by_index(i
            (device_name = device_info["name"]).lower
            if device_info["maxOutputChannels"] > 0 and ("headset" in device_name or
                :("speaker" in device_name or "headphones" in device_name
            :if device_info["name"] not in input_devices
                device_index = i
            :(if try_to_open_output_stream(device_index
                (["input_devices.append(device_info["name
                    ()p.terminate
                return input_devices

:()def get_input_devices
    [] = output_devices
    ()p = pyaudio.PyAudio
    :()for i in range(p.get_device_count
        (device_info = p.get_device_info_by_index(i

```

```

        ()device_name = device_info["name"].lower
:(if device_info["maxInputChannels"] > 0 and ("microphone" in device_name
        :if device_info["name"] not in output_devices
            device_index = i
        :(if try_to_open_input_stream(device_index
        (["output_devices.append(device_info["name
            ()p.terminate
        return output_devices

```

```

:(def set_icon_from_path_to_label(label, image_path

```

```

    Load the image from file path #

```

```

    (pixmap = QPixmap(image_path

```

```

        Get the size of the label #

```

```

        ()label_size = label.size

```

```

        ()label_width = label_size.width

```

```

        ()label_height = label_size.height

```

```

    Calculate the aspect ratio of the image #

```

```

        ()image_width = pixmap.width

```

```

        ()image_height = pixmap.height

```

```

    image_aspect_ratio = image_width / image_height

```

```

    Determine how to scale the image based on its aspect ratio #

```

```

        :if image_aspect_ratio <= 0.5

```

```

        scaled_pixmap = pixmap.scaledToWidth(label_width,
            (Qt.SmoothTransformation

```

```

        :elif image_aspect_ratio >= 1.5

```

```

scaled_pixmap = pixmap.scaledToHeight(label_height,
                                      (Qt.SmoothTransformation
                                      :else
scaled_pixmap = pixmap.scaled(label_size, Qt.KeepAspectRatio,
                              (Qt.SmoothTransformation

```

```

Set the scaled pixmap to the label #
    (label.setPixmap(scaled_pixmap
    (label.setAlignment(Qt.AlignCenter

```

```

:(def set_icon_from_bytes_to_label(label, image_bytes
    Load the image from bytes #
        ()pixmap = QPixmap
    (pixmap.loadFromData(image_bytes

```

```

    Get the size of the label #
        ()label_size = label.size
    ()label_width = label_size.width
    ()label_height = label_size.height

```

```

    Calculate the aspect ratio of the image #
        ()image_width = pixmap.width
        ()image_height = pixmap.height
    image_aspect_ratio = image_width / image_height

```

```

    Determine how to scale the image based on its aspect ratio #
        :if image_aspect_ratio <= 0.5

```

```

scaled_pixmap = pixmap.scaledToWidth(label_width,
                                     (Qt.SmoothTransformation
                                     :elif image_aspect_ratio >= 1.5
scaled_pixmap = pixmap.scaledToHeight(label_height,
                                     (Qt.SmoothTransformation
                                     :else
scaled_pixmap = pixmap.scaled(label_size, Qt.KeepAspectRatio,
                               (Qt.SmoothTransformation

```

```

Set the scaled pixmap to the label #
    (label.setPixmap(scaled_pixmap
    (label.setAlignment(Qt.AlignCenter

```

```

:(def set_button_icon(button, icon_path, width, height
    :try
        (icon = QIcon(icon_path
        (button.setIcon(icon
        (icon_size = QSize(width, height

```

```

Use the provided width and height to scale the icon #
    ()scaled_size = icon.pixmap(icon_size).size
    (button.setIconSize(scaled_size
    :except Exception as e
    ("{print(f"Error in setting button icon: {e

```

```

:(def extract_number(s
Use regular expression to find the number in the string #

```

```

        (match = re.search(r'\d+', s
                                :if match
Convert the matched number to an integer and return it #
        (())return int(match.group
                                :else
If no number is found, return None or raise an exception, depending on #
                                your use case
("return None # or raise ValueError("No number found in the string

```

```

:(def make_circular_image(image_bytes
.Converts an image to a circular image with the same width and height"""

```

```

:Args
.image_bytes (bytes): Image data as bytes

```

```

:Returns
.bytes: The circular image as bytes
"""

```

```

:try
    Load the image using Pillow #
:with BytesIO(image_bytes) as image_buffer
    (image = Image.open(image_buffer

```

```

(Convert the image to RGBA mode (if not already #
    :if image.mode != 'RGBA
    ('image = image.convert('RGBA

```

```

Determine the minimum dimension for the circular image #

```



```

(min_dimension = min(image.width, image.height

Create a square-shaped image #
square_image = Image.new('RGBA', (min_dimension, min_dimension),
                           (((255, 255, 255, 0
offset = ((min_dimension - image.width) // 2, (min_dimension -
                                                (image.height) // 2
(square_image.paste(image, offset

Determine the maximum circular area based on the minimum #
dimension
max_radius = min_dimension // 2

Create a mask with a transparent circle #
mask = Image.new('L', (min_dimension, min_dimension), 0) # Create a
black mask
(draw = ImageDraw.Draw(mask
(draw.ellipse(((0, 0), (min_dimension, min_dimension))), fill=255

Apply the mask to the image #
circular_image = Image.new('RGBA', (min_dimension, min_dimension),
                           (((255, 255, 255, 0
(circular_image.paste(square_image, mask=mask

Convert the circular image to bytes #
(output_buffer = BytesIO
circular_image.save(output_buffer, format='PNG') # Save as PNG format
(circular_image_bytes = output_buffer.getvalue

return circular_image_bytes

```

```

        :except Exception as e
        ("{print(f"Error converting image: {e
        return None

:(def make_q_object_clear(object
(";object.setStyleSheet("background-color: transparent; border: none

:()def get_camera_names
    [] = camera_names
for i in range(10): # You can adjust the range according to your needs
    (cap = cv2.VideoCapture(i
    :()if cap.isOpened
    ("{camera_names.append(f"Camera {i
    ()cap.release
    :else
    break
    return camera_names

:()def get_default_output_device_name
    ()p = pyaudio.PyAudio
    ()return p.get_default_output_device_info().get("name").lower

:()def get_default_input_device_name

```

```

        ()p = pyaudio.PyAudio
    ()return p.get_default_input_device_info().get("name").lower

```

```

:(def find_input_device_index(device_name
    ()p = pyaudio.PyAudio
    input_device_index = None

    :()for i in range(p.get_device_count
        (device_info = p.get_device_info_by_index(i
    :()if device_info["name"].lower() == device_name.lower
        input_device_index = i
        break

    ()p.terminate
    return input_device_index

```

```

:(def find_output_device_index(device_name
    ()p = pyaudio.PyAudio
    output_device_index = None

    :()for i in range(p.get_device_count
        (device_info = p.get_device_info_by_index(i
    :()if device_info["name"].lower() == device_name.lower
        output_device_index = i
        break

```

```

        ()p.terminate
    return output_device_index

class SettingsBox(QWidget):
    def __init__(self, parent):
        super().__init__()
        try:
            self.font_size = 60
            self.parent = parent
            self.Network = self.parent.Network
            self.settings_button_height = 50
            (self.file_dialog = QFileDialog(self
            (self.file_dialog.setFileMode(QFileDialog.ExistingFile
            ("self.file_dialog.setNameFilter("Image files (*.png *.jpg

            delta_of_main_buttons = 50
            starter_x_of_main_buttons = 350
            starter_y_of_main_buttons = 100

            (self.privacy_button_width, self.privacy_button_height = (200, 30

            label_height = 30
            label_width = 300
            self.default_labels_font_size = 10
            user_settings_label =
            self.create_white_label(starter_x_of_main_buttons+10 ,
            starter_y_of_main_buttons-35, self.default_labels_font_size,label_width,
            ("label_height, "USER SETTINGS

```

```

self.my_account_button = self.create_settings_main_buttons("My
                    ) ,Account", self.my_account_pressed
                    ((starter_x_of_main_buttons, starter_y_of_main_buttons

                    starter_y_of_main_buttons += delta_of_main_buttons

self.user_profile_button = self.create_settings_main_buttons("User
                    ) ,Profile", self.user_profile_pressed
                    ((starter_x_of_main_buttons, starter_y_of_main_buttons

                    starter_y_of_main_buttons += delta_of_main_buttons

                    self.appearance_button =
) ,self.create_settings_main_buttons("Appearance", self.appearance_pressed
                    ((starter_x_of_main_buttons, starter_y_of_main_buttons

                    starter_y_of_main_buttons += delta_of_main_buttons

self.voice_video_button = self.create_settings_main_buttons("Voice &&
                    ) ,Video", self.voice_video_pressed
                    ((starter_x_of_main_buttons, starter_y_of_main_buttons

                    starter_y_of_main_buttons += delta_of_main_buttons

self.privacy_safety_button = self.create_settings_main_buttons("Privacy
                    ) ,&& Safety", self.privacy_safety
                    ((starter_x_of_main_buttons, starter_y_of_main_buttons

                    starter_y_of_main_buttons += delta_of_main_buttons

```

```

,"self.log_out_button = self.create_settings_main_buttons("Log Out

                                ),self.parent.page_controller_object.log_out
                                ((starter_x_of_main_buttons, starter_y_of_main_buttons

                                background_color = self.parent.background_color_hex
                                hover_color = self.parent.standard_hover_color

                                (self.label = QLabel(self
                                self.label.setStyleSheet(f"border-right: 3px solid
                                (";{self.parent.standard_hover_color}; padding-left: 10px
                                self.label.setGeometry(starter_x_of_main_buttons +
                                (self.privacy_safety_button.width()-3, -20, 3, 1020

                                """" = self.combo_box_style_sheet
                                } QComboBox
                                ;background-color: %s
                                ;selection-background-color: %s
                                ;border: 1px solid %s
                                ;border-radius: 5px
                                ;padding: 2px 18px 2px 3px
                                ;color: white
                                /* min-width: 150px; /* Adjust min-width as needed
                                /* max-width: 500px; /* Set max-width to accommodate longer text
                                /* font-size: 14px; /* Adjust font size as needed
                                {

                                } QComboBox::drop-down
                                ;subcontrol-origin: padding

```

```

subcontrol-position: top right; /* Position the drop-down at the top
/* right
;width: 20px
;border-left: 1px solid transparent
{

} QComboBox QAbstractItemView
;color: white
;background-color: %s
;selection-background-color: %s
;padding: 2px
;font-size: 14px
{

background_color, hover_color, hover_color, background_color,) % ""
(hover_color

"slider_style_sheet_color = "#3498db
""self.volume_slider_style_sheet = f
}} QSlider::groove:horizontal
;border: 1px solid #bbb
background: qlineargradient(x1:0, y1:0, x2:1, y2:0, stop:0
;(#ddd, stop:1 #eee
;height: 10px
;margin: 0px
{{

}} QSlider::handle:horizontal
background: qlineargradient(x1:0, y1:0, x2:1, y2:1, stop:0
;(#eee, stop:1 #ccc

```

```

        ;border: 1px solid #777
        ;width: 20px
margin: -2px 0; /* handle is placed by default on the contents
        /* rect of the groove. Expand outside the groove
        ;border-radius: 5px
        {{

    }} QSlider::add-page:horizontal
        ;background: #fff
        {{

    }} QSlider::sub-page:horizontal
background: {slider_style_sheet_color}; /* Change this color
        /* to the desired color for the left side
        {{
        ""

```

```

label_page = self.create_white_label(800, 70, 20, None, None,
                                     (self.parent.selected_settings
                                     :except Exception as e
                                     ("{print(f"error in creating setting box {e
                                     :try
                                     "dark_green = "#1e9644
                                     "other_green = "#044f1c
                                     "red_hex = "#9e0817
                                     "dark_red_hex = "#690d16
                                     :if self.parent.selected_settings == "My Account
                                     start_y = 100
                                     start_x = 500

```



```

        "dark_green = "#1e9644
        "other_green = "#044f1c
        (width, height = (120, 120
self.profile_image_label = create_custom_circular_label(width, height,
                                                                    (self
                                                                    (profile_image_x, profile_image_y = (800, 200
                                                                    user_image =
        (self.parent.get_circular_image_bytes_by_name(self.parent.username
                                                                    :if user_image is None
        "icon_path = "discord_app_assets/regular_profile.png
        (set_icon_from_path_to_label(self.profile_image_label, icon_path
                                                                    :else
        (set_icon_from_bytes_to_label(self.profile_image_label, user_image
        (self.profile_image_label.move(profile_image_x, profile_image_y

label_name_next_to_image_x, label_name_next_to_image_y = (950,
                                                                    (240
                                                                    label_name_next_to_image =
        self.create_white_label(label_name_next_to_image_x,
        (label_name_next_to_image_y, 20, None, None, self.parent.username

        (button_edit_user_profile_x, button_edit_user_profile_y = (1250, 240
        (button_width, button_height = (180, 50
        button_edit_user_profile = self.create_colored_button(dark_green,
        other_green, None, button_edit_user_profile_x, button_edit_user_profile_y,
        ("button_width", button_height, "Edit User Profile
        (button_edit_user_profile.clicked.connect(self.user_profile_pressed

        first_account_label_y = label_name_next_to_image_y+120
        account_label_x = label_name_next_to_image_x-135

```

```

self.create_my_account_labels(account_label_x, first_account_label_y,
    (self.default_labels_font_size+2, "USERNAME", self.parent.username
        y = first_account_label_y + 80
        text = self.parent.email
        :if text is None
        "text = "None
    self.create_my_account_labels(account_label_x, y,
        ,self.default_labels_font_size+2
    (EMAIL", text"
        y = y + 80
        text = self.parent.phone_number
        :if text is None
        "text = "None
    self.create_my_account_labels(account_label_x, y,
        ,self.default_labels_font_size+2
    (PHONE NUMBER", text"
    change_password_button_x, change_password_button_y =
        account_label_x-30, y+120
    change_password_button = self.create_colored_button(dark_green,
        ,other_green, None
    ,change_password_button_x
    change_password_button_y,
        ,button_width
    ("button_height, "Change Password
        delete_account_button_x, delete_account_button_y =
            change_password_button_x, change_password_button_y+100
    delete_account_button = self.create_colored_button(red_hex,
        ,dark_red_hex, None
    ,delete_account_button_x
    delete_account_button_y,
        ,button_width

```

```
("button_height, "Delete Account
```

```
:"elif self.parent.selected_settings == "Voice & Video
```

```
    Check if the device has input capability #
```

```
        ()input_devices = get_input_devices
```

```
        ()output_devices = get_output_devices
```

```
camera_names_list = self.parent.camera_devices_names
```

```
        starter_y = 170
```

```
        volume_slider_y = starter_y+100
```

```
        volume_slider_label_y = volume_slider_y - 15
```

```
        volume_slider__x = 800
```

```
        volume_slider_label_x = volume_slider__x
```

```
        (width, height = (300, 45
```

```
            slider_min_value = 0
```

```
            slider_max_value = 100
```

```
        self.volume_slider = create_slider(self, slider_min_value,  
slider_max_value, self.parent.volume, self.set_volume, volume_slider__x,  
        volume_slider_y
```

```
        width, height, ,
```

```
        (self.volume_slider_style_sheet
```

```
volume_slider_label = self.create_white_label(volume_slider_label_x,  
        volume_slider_label_y, self.default_labels_font_size, None, None, "OUTPUT  
        ("VOLUME
```

```
self.volume_label = self.create_white_label(volume_slider_label_x +  
        width + 10, volume_slider_y+7, self.default_labels_font_size, 100, 30,  
        ((str(self.parent.volume
```

```
space_between_option_box_and_label = 30
```

```

        (output_x, output_y = (800, starter_y
self.output_combobox = self.create_option_box(width, height,
        (output_x, output_y, output_devices
        ("self.output_combobox.addItem("Default

self.output_combobox.currentIndexChanged.connect(self.output_device_change
        (d

        output_label = self.create_white_label(output_x,
output_y-space_between_option_box_and_label, self.default_labels_font_size,
        ("None, None, "OUTPUT DEVICES

        :if self.parent.output_device_name not in output_devices
        ("self.output_combobox.setCurrentText("Default

        (input_x, input_y = (1150, starter_y
self.input_combobox = self.create_option_box(width, height, input_x,
        (input_y, input_devices
        ("self.input_combobox.addItem("Default

(self.input_combobox.currentIndexChanged.connect(self.input_device_changed
        :if self.parent.input_device_name not in input_devices
        ("self.input_combobox.setCurrentText("Default

        input_label = self.create_white_label(input_x, input_y -
,space_between_option_box_and_label, self.default_labels_font_size, None
        ("None, "INPUT DEVICES

        (camera_x, camera_y = (800, 670
self.camara_devices_combobox = self.create_option_box(width,
        (height, camera_x, camera_y, camera_names_list

self.camara_devices_combobox.currentIndexChanged.connect(self.camera_devi
        (ce_changed

```

```

        camera_label = self.create_white_label(camera_x, camera_y -
,space_between_option_box_and_label, self.default_labels_font_size, None
        ("None, "CAMERA
            :if self.parent.camera_index is None
self.camara_devices_combobox.setCurrentText("No Devices
                                                    ("Found
                                                    :else

((self.camara_devices_combobox.setCurrentText(str(self.parent.camera_index
        (input_mode_x, input_mode_y = (800, 370
(self.create_input_mode_select_button(input_mode_y, input_mode_x
        input_mode_label = self.create_white_label(input_mode_x,
        input_mode_y - space_between_option_box_and_label,
        ,self.default_labels_font_size, None
        ("None, "INPUT MODE
        (push_to_talk_select_x, push_to_talk_select_y = (800, 550
            push_to_talk_select =
        self.create_select_push_to_talk_key(push_to_talk_select_x,
            (push_to_talk_select_y
            push_to_talk_select_label =
        self.create_white_label(push_to_talk_select_x, push_to_talk_select_y -
,space_between_option_box_and_label, self.default_labels_font_size, None
        ("None, "Push to Talk Keybind

        :elif self.parent.selected_settings == "Appearance
            starter_y = 170
            space_between_option_box_and_label = 30
            list_optional_colors = self.parent.color_design_options
            (width, height = (300, 45
        appearance_select_box_x, appearance_select_box_y = (800,
                                                    (starter_y

```

```

        self.color_combobox = self.create_option_box(width, height,
        (appearance_select_box_x, appearance_select_box_y, list_optional_colors
        (self.color_combobox.setCurrentText(self.parent.background_color

self.color_combobox.currentIndexChanged.connect(self.background_color_chan
        (ged

        appearance_select_box_label =
self.create_white_label(appearance_select_box_x, appearance_select_box_y -
        ,space_between_option_box_and_label, self.default_labels_font_size, None
        ("None, "THEME COLOR

        font_size_slider_x, font_size_slider_y = 800, starter_y+100
        font_size_slider_style_sheet = self.volume_slider_style_sheet
        (font_size_slider_width, font_size_slider_height = (300, 50
        (font_size_slider_min_value, font_size_slider_max_value = (6, 20

        font_slider_label = self.create_white_label(font_size_slider_x,
        ,font_size_slider_y-15
self.default_labels_font_size, None, None,

        (""FONT SIZE

        self.font_size_slider = create_slider(self, font_size_slider_min_value,
        ,font_size_slider_max_value, self.parent.font_size

        self.font_size_changed, font_size_slider_x,

        font_size_slider_y

        font_size_slider_width, font_size_slider_height, ,
        (font_size_slider_style_sheet

        self.font_size_label = self.create_white_label(font_size_slider_x +
        ,font_size_slider_width + 10, font_size_slider_y + 7
        ,self.default_labels_font_size, 100, 30
        ((str(self.parent.font_size

        font_option_x, font_option_y = 800, starter_y+200

```

```

        font_slider_label = self.create_white_label(font_option_x,
                                                    ,font_option_y-space_between_option_box_and_label
self.default_labels_font_size, None, None,
                                                    (""FONT STYLE

self.font_box = self.create_option_box(width, height, font_option_x,
                                        (font_option_y, self.parent.font_options
                                        (self.font_box.setCurrentText(self.parent.font_text
                                        (self.font_box.currentIndexChanged.connect(self.font_updated

        :elif self.parent.selected_settings == "User Profile
            (width, height = (120, 120
self.profile_image_label = create_custom_circular_label(width, height,
                                                    (self
            """)self.profile_image_label.setStyleSheet
            } QLabel
border-radius: "" + str(height // 2) + ""px; /* Set to half of the label
                                                    /* height
            {
            (""

        (profile_image_x, profile_image_y = (800, 200
            user_image =
        (self.parent.get_circular_image_bytes_by_name(self.parent.username
            :try
            :if user_image is None
            "icon_path = "discord_app_assets/regular_profile.png
        (set_icon_from_path_to_label(self.profile_image_label, icon_path
            :else
        set_icon_from_bytes_to_label(self.profile_image_label,
                                                    (user_image
            :except Exception as e

```

```

        ({print(f"error in putting profile image on label {e
(self.profile_image_label.move(profile_image_x, profile_image_y
(width_change_profile_pic, height_change_profile_pic = (160, 40
        (x_change_profile_pic, y_change_profile_pic = (800, 400
change_profile_pic_button = self.create_colored_button(dark_green,
                                                ,other_green, None
        ,x_change_profile_pic
        y_change_profile_pic,
                                                ,width_change_profile_pic
height_change_profile_pic, "Change
                                                ("Avatar

(change_profile_pic_button.clicked.connect(self.edit_profile_pic_pressed
        (x_remove_profile_pic, y_remove_profile_pic = (1000, 400
remove_profile_pic_button = self.create_colored_button(red_hex,
                                                ,dark_red_hex, None
        ,x_remove_profile_pic
        y_remove_profile_pic,
                                                ,width_change_profile_pic
height_change_profile_pic, "Remove
                                                ("Avatar

(remove_profile_pic_button.clicked.connect(self.remove_profile_pic
        : "elif self.parent.selected_settings == "Privacy & Safety
        (privacy_page_starter_x, privacy_page_starter_y = (800, 150
        space_between_labels = 120
option_list = ["Private account", "Censor data from strangers", "Two
                                                ["factor authentication

        self.create_privacy_labels(privacy_page_starter_x,
        (privacy_page_starter_y, option_list, space_between_labels
        button_starter_x, button_starter_y = privacy_page_starter_x,
        privacy_page_starter_y + 50

```



```

        labels_matching_vars_list = [self.parent.is_private_account,
[self.parent.censor_data_from_strangers, self.parent.two_factor_authentication
, "vars_names = ["is_private_account", "censor_data_from_strangers
                ["two_factor_authentication"

self.create_privacy_buttons(button_starter_x, button_starter_y,
                (space_between_labels, labels_matching_vars_list, vars_names

                                :except Exception as e

                                ("){print(f"error setting page {e

                                :

                                : (def input_device_changed(self
()self.parent.input_device_name = self.input_combobox.currentText
                ("){print(f"changed input device to {self.parent.input_device_name
                                :()if self.parent.play_vc_data_thread.is_alive
                                ()self.parent.close_send_vc_thread
                                ()self.parent.start_send_vc_thread

                                : (def output_device_changed(self
()self.parent.output_device_name = self.output_combobox.currentText
                ("){print(f"changed output device to {self.parent.output_device_name
                                :()if self.parent.play_vc_data_thread.is_alive
                                ()self.parent.close_listen_thread
                                ()self.parent.start_listen_thread

                                : (def camera_device_changed(self
                                self.parent.camera_index =
                                (()extract_number(self.camara_devices_combobox.currentText
                ("){print(f"changed camera decive index to {self.parent.camera_index
                                :()if self.parent.send_camera_data_thread.is_alive
                                ()self.parent.end_share_camera_thread

```

```

        ()self.parent.start_camera_data_thread

def create_privacy_labels(self, starter_x, starter_y, list_of_label_content,
                        :(space_between_labels
                        :for content in list_of_label_content
                        ,label1 = self.create_white_label(starter_x, starter_y
self.default_labels_font_size, None, None,
                        (content
                        starter_y += space_between_labels

def create_privacy_buttons(self, starter_x, starter_y, space_between,
                        :(list_of_button_vars, var_names
                        "off_icon_path = "discord_app_assets/off_button.png
                        "on_icon_path = "discord_app_assets/on_button.png
                        index = 0
                        for index, (var, var_name) in enumerate(zip(list_of_button_vars,
                        :(var_names
                        (button = QPushButton(self
                        (make_q_object_clear(button
button.setFixedSize(button_width, button_height) # Set the size of the #
                        button
                        :if var
                        set_button_icon(button, on_icon_path, self.privacy_button_width,
                        (self.privacy_button_height
                        :else
                        set_button_icon(button, off_icon_path, self.privacy_button_width,
                        (self.privacy_button_height
                        (button.move(starter_x, starter_y
                        )button.clicked.connect
                        lambda checked, btn=button, name=var_name:
                        (self.switch_off_variable_plus_change_icon(btn, name

```

```

(
    starter_y += space_between

:(def switch_off_variable_plus_change_icon(self, button, var_name
    "off_icon_path = "discord_app_assets/off_button.png
    "on_icon_path = "discord_app_assets/on_button.png
    :try

        (var_value = not getattr(self.parent, var_name
set_button_icon(button, on_icon_path if var_value else off_icon_path,
                    ,self.privacy_button_width
                    (self.privacy_button_height
                    (setattr(self.parent, var_name, var_value
                        :except Exception as e
                        ("){print(f"error in changing privacy button icon {e

:(def background_color_changed(self
    ()new_background_color = self.color_combobox.currentText
    (self.parent.update_background_color(new_background_color
        (print(new_background_color

:(def font_updated(self
    ()new_font = self.font_box.currentText
    self.parent.font_text = new_font
    ("){print(f"new_font is {new_font

:(def remove_profile_pic(self
    :try

    self.parent.profile_pic = None

    :try

```

```

        (self.Network.send_profile_pic(None
                                     :except Exception as e
        ("{print(f"error in sending profile picture: {e
        ("print("send profile pic to server
        ()self.parent.activateWindow
(self.parent.update_profile_pic_dicts_list(self.parent.username, None
        ()self.parent.updated_settings_page
        :except Exception as e
        ("{print(f"error in resetting profile pic {e

        : (def edit_profile_pic_pressed(self
        ()self.open_file_dialog

        : (def open_file_dialog(self
        : ()_if self.file_dialog.exec
        ()selected_files = self.file_dialog.selectedFiles
        :if selected_files
        [file_path = selected_files[0
        ([image_bytes = file_to_bytes(selected_files[0
        : (if is_valid_image(image_bytes
        self.parent.profile_pic = image_bytes
        (self.Network.send_profile_pic(image_bytes
        ("print("send profile pic to server
        ()self.parent.activateWindow
        (circular_pic = make_circular_image(image_bytes
        :if circular_pic is not None
        set_icon_from_bytes_to_label(self.profile_image_label,
                                     (circular_pic

```

```

self.parent.update_profile_pic_dicts_list(self.parent.username,
                                          (image_bytes, circular_pic

:(def font_size_changed(self, font_size
                          :try
                          (self.parent.font_size = int(font_size
((self.font_size_label.setText(str(font_size
                          :except Exception as e
("){print(f"font_size_changed error :{e

:(def create_my_account_labels(self, x, y, font_size, text1, text2
label1 = self.create_custom_label(x, y, font_size, None, None, text1, "grey",
                                  (False
label2 = self.create_custom_label(x, y+font_size+15, font_size, None, None,
                                  (text2, "white", False

:(def change_input_mode(self
:if self.parent.is_push_to_talk
self.parent.is_push_to_talk = False
:else
self.parent.is_push_to_talk = True
()self.parent.updated_settings_page

def create_colored_button(self, background_color, hover_color, border_color,
                          :(x, y, width, height, text
(new_button = QPushButton(text, self
:if border_color is None
"border_color = "#00000000
""new_button.setStyleSheet(f
}} QPushButton

```

```

        ;{background-color: {background_color
            ;{border: 2px solid {border_color
                ;border-radius: 5px
                ;padding: 8px 16px
/* padding-left: 35px; /* Adjust the padding to move text to the right
                ;color: white
                ;font-family: Arial, sans-serif
                ;font-size: 14px
                ;font-weight: normal
            ;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
                ;text-align: left
                {{
                }} QPushButton: hover
            ;{background-color: {hover_color
                {{
                (""""
(new_button.setGeometry(x, y, width, height

return new_button

:(def create_select_push_to_talk_key(self, x, y
    "red = "#FF0000
    :if self.parent.is_editing_push_to_talk_button
push_to_talk_label = self.push_to_talk_label(x, y, 20, 340, 50,
    (self.parent.push_to_talk_key, "red", red
    :else
        border_color = self.parent.standard_hover_color
    :if self.parent.background_color == "Black and White
        "font_color = "black

```

```

:else

        "font_color = "white

push_to_talk_label = self.push_to_talk_label(x, y, 20, 340, 50,
        (self.parent.push_to_talk_key, font_color, border_color

        hover_red = "#CC0000" # Slightly darker than red

        "grey = "#808080
        hover_grey = "#6e6e6e" # slightly darker than grey

        button_width = 160
        button_height = 37

        :if self.parent.is_editing_push_to_talk_button
record_keybind_button = self.create_colored_button(red, hover_red, red,
        ("x + 170, y+6, button_width, button_height, "Stop Recording

record_keybind_button.clicked.connect(self.handle_push_to_talk_selection_button_
        (n_clicked

        :else

edit_keybind_button = self.create_colored_button(grey, hover_grey, grey,
        ("x+170, y+6, button_width, button_height, "Edit Keybind

edit_keybind_button.clicked.connect(self.handle_push_to_talk_selection_button_
        (clicked

:(def handle_push_to_talk_selection_button_clicked(self
        :if self.parent.is_editing_push_to_talk_button
self.parent.is_editing_push_to_talk_button = False

        :else

self.parent.is_editing_push_to_talk_button = True

```

```

        ()self.parent.updated_settings_page

def push_to_talk_label(self, x, y, font_size, width, height, text, color,
                        border_color):
    brighter_blue = self.parent.standard_hover_color
    if text is None:
        label = QLabel("None", self)
    else:
        label = QLabel(text, self)
    if width is None and height is None:
        label.move(x, y)
    else:
        label.setGeometry(x, y, width, height)

    label.setStyleSheet(f"""
        {{
            QLabel
            ;{{background-color: {brighter_blue}
            ;{{border: 2px solid {border_color}
            ;border-radius: 5px
            ;padding: 8px 16px
            ;{{color: {color}
            ;font-family: Arial, sans-serif
            ;font-size: {font_size}px
            ;font-weight: normal
            ;text-align: left
            {{
            ("""

```



```

return label

:(def create_input_mode_select_button(self, starter_y, buttons_x
                                     "regular_blue = "#192549
                                     brighter_blue = self.parent.standard_hover_color
                                     width_buttons = 620
                                     height_buttons = 50
                                     "selected_path = "discord_app_assets/select_circle.png
                                     "not_selected_path = "discord_app_assets/not_select_circle.png
                                     icons_size = 30
                                     "no_background_color = "transparent
                                     border_color = self.parent.standard_hover_color
                                     :if not self.parent.is_push_to_talk
                                     "text1 = "Voice Activity
voice_activity_button = self.create_colored_button(no_background_color,
  brighter_blue, border_color, buttons_x, starter_y, width_buttons,height_buttons,
                                                    (text1
                                                    "text2 = "Push to Talk
second_button_y = starter_y + 60
push_to_talk_button = self.create_colored_button(no_background_color,
  brighter_blue, border_color, buttons_x, second_button_y,
  (width_buttons,height_buttons, text2
  (voice_activity_button.clicked.connect(self.change_input_mode
  (push_to_talk_button.clicked.connect(self.change_input_mode

  selected_button_image = self.create_image_label(selected_path,
    (icons_size, icons_size, buttons_x + 5, starter_y+10
    not_selected_button_image =
self.create_image_label(not_selected_path, icons_size, icons_size, buttons_x +
  (5, second_button_y + 10

```

```

:else

        "text1 = "Voice Activity

voice_activity_button = self.create_colored_button(no_background_color,
    brighter_blue, border_color, buttons_x, starter_y, width_buttons, height_buttons,
    (text1

        "text2 = "Push to Talk

        second_button_y = starter_y + 60

push_to_talk_button = self.create_colored_button(no_background_color,
    brighter_blue, border_color, buttons_x, second_button_y,
    (width_buttons,height_buttons, text2

    (voice_activity_button.clicked.connect(self.change_input_mode
    (push_to_talk_button.clicked.connect(self.change_input_mode

    selected_button_image = self.create_image_label(selected_path,
        (icons_size, icons_size, buttons_x + 5, second_button_y + 10

        not_selected_button_image =
self.create_image_label(not_selected_path, icons_size, icons_size, buttons_x +
    (5, starter_y + 10

        : "if self.parent.background_color == "Black and White

        push_to_talk_button.setStyleSheet(push_to_talk_button.styleSheet() +
            (";color: black

voice_activity_button.setStyleSheet(voice_activity_button.styleSheet() +
    (";color: black

:(def create_option_box(self, width, height, x, y, item_list
    (color_combobox = CustomComboBox(self

        :if len(item_list) > 0

        :for i in item_list

        (color_combobox.addItem(i

        :else

```

```

        ("color_combobox.addItem("No Devices Found

(color_combobox.setStyleSheet(self.combo_box_style_sheet

        (color_combobox.setGeometry(x, y, width, height
        "icon_path = "discord_app_assets/down_arrow_icon.png
arrow_label = self.create_image_label(icon_path, 30, x + (width * 0.87), x +
                                         ((width * 0.87), y + 10
color_combobox.visibility_changed.connect(lambda is_visible:
    ((handle_combobox_visibility_changed(is_visible, arrow_label
        arrow_label.mousePressEvent = lambda event:
            ()color_combobox.showPopup

        return color_combobox

:(def create_image_label(self, image_path, height, width, x, y
    (image_label = QLabel(self
(";image_label.setStyleSheet("background-color: transparent
        icon_path = image_path
        (button_icon = QIcon(icon_path
        (pixmap = button_icon.pixmap(width, height
        (image_label.setPixmap(pixmap
        (image_label.move(x, y
        return image_label

:(def create_white_label(self, x, y, font_size, width, height, text
white_label = self.create_custom_label(x, y, font_size, width, height, text,
                                         ("white", True
        return white_label

```

```

def create_custom_label(self, x, y, font_size, width, height, text, color=None,
                        :is_bold=False
                        (custom_label = QLabel(text, self
                        :if width is None and height is None
                        (custom_label.move(x, y
                        :else
                        (custom_label.setGeometry(x, y, width, height

                        Set text color #
                        :if color is None
                        color = "white" # Default color is white if not specified
                        custom_label.setStyleSheet("color: {}; font-size: {}pt;".format(color,
                                                ((font_size

                        Set text weight to bold if specified #
                        :if is_bold
                        custom_label.setStyleSheet(custom_label.styleSheet() + "font-weight:
                                                (";bold

                        return custom_label

                        :(def change_username_function(self
                        Implement the function for changing the username #
                        pass

                        :(def change_password_function(self
                        Implement the function for changing the password #
                        pass

```

```

:(def delete_account_function(self
Implement the function for deleting the account #
pass

```

```

:(def set_volume(self, value
self.parent.volume = value
((self.volume_label.setText(str(value
(self.parent.update_media_players_volume(value

```

```

:(def my_account_pressed(self
:"if self.parent.selected_settings != "My Account
"self.parent.selected_settings = "My Account
()self.parent.updated_settings_page

```

```

:(def user_profile_pressed(self
:"if self.parent.selected_settings != "User Profile
"self.parent.selected_settings = "User Profile
()self.parent.updated_settings_page

```

```

:(def appearance_pressed(self
:"if self.parent.selected_settings != "Appearance
"self.parent.selected_settings = "Appearance
()self.parent.updated_settings_page

```

```

:(def voice_video_pressed(self
:"if self.parent.selected_settings != "Voice & Video
"self.parent.selected_settings = "Voice & Video
()self.parent.updated_settings_page

```

```

                                : (def privacy_safety(self
: "if self.parent.selected_settings != "Privacy & Safety
    "self.parent.selected_settings = "Privacy & Safety
                                ()self.parent.updated_settings_page

: (def create_settings_main_buttons(self, text, funcion, position

                                button_text = text

                                (button = QPushButton(self
                                (button.setText(button_text
                                ([button.move(position[0], position[1
                                (button.setFixedHeight(self.settings_button_height
                                (button.clicked.connect(funcion

                                ""button.setStyleSheet(f
                                }} QPushButton
;{background-color: {self.parent.background_color_hex
    ;{border: 2px solid {self.parent.standard_hover_color
                                ;border-radius: 5px
                                ;padding: 8px 16px
/* padding-left: 35px; /* Adjust the padding to move text to the right
                                ;color: white
                                ;font-family: Arial, sans-serif
                                ;font-size: 14px
                                ;font-weight: normal
                                ;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1

```

```

        ;text-align: left
        {{

    }} QPushButton: hover
    ;{background-color: {self.parent.standard_hover_color
        {{

    }} QPushButton: pressed
    ;background-color: #202225
    ;border-color: #72767d
        {{
        ("""

(button.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Fixed
    (button.setFixedWidth(350
        )_button.raise

    return button

def generate_button_stylesheet(self, normal_color, hover_color,
                                :(pressed_color
                                """return f

    }} QPushButton
    ;{background-color: {normal_color
    ;{border: 2px solid {self.parent.standard_hover_color
        ;border-radius: 5px
        ;padding: 8px 16px
/* padding-left: 35px; /* Adjust the padding to move text to the right
        ;color: white

```

```

;font-family: Arial, sans-serif
;font-size: 14px
;font-weight: normal
;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
;text-align: left
{{

}} QPushButton:hover
;{background-color: {hover_color
{{

}} QPushButton:pressed
;{background-color: {pressed_color
;border-color: #72767d
{{
      ""

```

```

:(class CustomComboBox(QComboBox
(visibility_changed = pyqtSignal(bool

:(def __init__(self, parent=None
(super(CustomComboBox, self).__init__(parent

:(def showPopup(self
)super(CustomComboBox, self).showPopup
(self.visibility_changed.emit(True

```



```

        : (def hidePopup(self
()super(CustomComboBox, self).hidePopup
        (self.visibility_changed.emit(False

from youtube_search import YoutubeSearch
        from pytube import YouTube
            import tempfile
                import os
                    import moviepy.editor as mp
                        from PIL import Image
                            from io import BytesIO
                                import requests
                                    import random
                                        import string

:(def generate_random_filename(length=10
    chars = string.ascii_letters + string.digits
    ((return "".join(random.choice(chars) for _ in range(length

:(def extract_audio_bytes(query
    :try
        Search YouTube for videos based on the query #
()search_results = YoutubeSearch(query, max_results=10).to_dict
        :if search_results
            Get the first video ID if available #
            :for result in search_results

```

```

        first_result = result
        (" ,video_id = first_result.get('id

        :if video_id

        Construct the video link #
        '{video_link = f'https://www.youtube.com/watch?v={video_id

        Get video stream #
        (yt = YouTube(video_link

        (if yt.length <= 420: # 420 seconds = 7 minutes (7 * 60

            Download the video #

            ()temp_path = tempfile.gettempdir

            "random_filename = generate_random_filename() + ".mp4
            ('{audio_path = os.path.join(temp_path, f'{random_filename

        yt.streams.filter(only_audio=True).first().download(output_path=temp_path,
            (filename=random_filename

        (Convert the downloaded video to audio (MP3 #
        (audio_clip = mp.AudioFileClip(audio_path

        (('audio_clip.write_audiofile(audio_path.replace('.mp4', '.mp3

        Read the audio file bytes #

        :with open(audio_path.replace('.mp4', '.mp3'), 'rb') as audio_file

            ()audio_bytes = audio_file.read

        Get video title and thumbnail bytes #

        video_title = yt.title

        (thumbnail_bytes = get_thumbnail_bytes(yt.thumbnail_url

```

```

        audio_duration = yt.length
duration_min_sec = f'{audio_duration // 60}:{audio_duration %
                                                                    "{60:02

```

```

        Clean up temporary files #
        (os.remove(audio_path
        (('os.remove(audio_path.replace('.mp4', '.mp3

        (!print("Audio bytes extracted successfully
return audio_bytes, video_title, thumbnail_bytes,
                                                                    duration_min_sec
                                                                    :else
        (".print("No video ID found for the search query
                                                                    :else
        (".print("No videos found for the given query
                                                                    :except Exception as e
        ("{print(f"An error occurred: {e
        return None, None, None, None

```

```

        :(def get_thumbnail_bytes(thumbnail_url
        (response = requests.get(thumbnail_url
        :if response.status_code == 200
        ((img = Image.open(BytesIO(response.content
        (img_bytes = BytesIO
        ('img.save(img_bytes, format='PNG
        (return img_bytes.getvalue
                                                                    :else
        return None

```

```

import socket
import json
import zlib
import threading
import logging
import base64

from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import padding as aes_padding
import secrets
import pickle

'vc_data_sequence = br'\vc_data
'share_screen_sequence = br'\share_screen_data
'share_camera_sequence = br'\share_camera_data


:(def slice_up_data(data, mtu
    [] = sliced_data
    :(for start_index in range(0, len(data), mtu
        end_index = start_index + mtu
        ([sliced_data.append(data[start_index:end_index
            return sliced_data

```

```

:(def create_dictionary_with_message_type(message_type, keys, values
      Ensure both lists have the same length #
      :(if len(keys) != len(values)
        ("raise ValueError("Lists must have the same length

      Add the message_type as the first key-value pair #
      ('keys.insert(0, 'message_type
      (values.insert(0, message_type

      Create the dictionary using a dictionary comprehension #
      {((result = {keys[i]: values[i] for i in range(len(keys)

      return result

```

```

:()def generate_secure_symmetric_key
(symmetric_key = secrets.token_bytes(32
  return symmetric_key

```

```

:()def generate_aes_key
Generate a random 256-bit (32-byte) key for AES-256 #
(aes_key = secrets.token_bytes(32
  return aes_key

```

```

:()def generate_rsa_key_pair
(private_key = rsa.generate_private_key

```

```

        ,public_exponent=65537
        ,key_size=2048
        ()backend=default_backend
    )
    return private_key.public_key(), private_key

:(def encrypt_with_rsa(public_key, data
    )ciphertext = public_key.encrypt
        ,data
        )padding.OAEP
,((()mgf=padding.MGF1(algorithm=hashes.SHA256
    ,()algorithm=hashes.SHA256
        label=None
        (
        (
("return base64.b64encode(ciphertext).decode("utf-8

:(def decrypt_with_rsa(private_key, ciphertext
    (ciphertext = base64.b64decode(ciphertext
        )plaintext = private_key.decrypt
            ,ciphertext
            )padding.OAEP
,((()mgf=padding.MGF1(algorithm=hashes.SHA256
    ,()algorithm=hashes.SHA256
        label=None
        (

```

```

        (
        return plaintext

    : (def encrypt_with_aes(key, data
cipher = Cipher(algorithms.AES(key), modes.ECB(),
                (())backend=default_backend
                (())encryptor = cipher.encryptor

        Use PKCS#7 padding #
        (())padder = aes_padding.PKCS7(128).padder
        (())padded_data = padder.update(data) + padder.finalize

    (())ciphertext = encryptor.update(padded_data) + encryptor.finalize
        (return base64.b64encode(ciphertext

    : (def decrypt_with_aes(key, ciphertext
        :try
            get ciphertext as bytes #
            (ciphertext = base64.b64decode(ciphertext
            cipher = Cipher(algorithms.AES(key), modes.ECB(),
                (())backend=default_backend
                (())decryptor = cipher.decryptor

            Decrypt the ciphertext #
    (())decrypted_data = decryptor.update(ciphertext) + decryptor.finalize
        Use PKCS#7 unpadding #
        (())unpadder = aes_padding.PKCS7(128).unpadder

```

```

()unpadded_data = unpadder.update(decrypted_data) + unpadder.finalize
        return type bytes #
    return unpadded_data
    :except Exception as e
        ("{print(f"Error in decryption: {e
            return 1

:(def send_data_in_chunks(sock, data
    .Send data over a socket in chunks""

    :Args

    .sock (socket.socket): The socket object for sending data
    .data (bytes): The data to be sent

    :Returns

    .bool: True if the data was sent successfully, False otherwise
        ""

    :try

        Define the chunk size #
        chunk_size = 4096 # Adjust this based on your requirements

        Send data in chunks #
        bytes_sent = 0
        :(while bytes_sent < len(data
            [chunk = data[bytes_sent:bytes_sent + chunk_size
                (bytes_sent += sock.send(chunk

```



```
        return True
```

```
    except Exception as e
```

```
        ("{print(f'Error sending data: {e
```

```
        return False
```

```
class ServerNet
```

```
    def __init__(self, s, addr
```

```
        self.client_tcp_socket_address = addr
```

```
        __self.logger = logging.getLogger(__name
```

```
        (self.logger.setLevel(logging.DEBUG
```

Create a StreamHandler with the desired format #

```
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -  
                                ('%(message)s
```

```
    stream_handler = logging.StreamHandler
```

```
    (stream_handler.setFormatter(formatter
```

```
        self.server = s
```

```
        self.size = 0000000
```

```
        self.original_len = 10
```

```
        self.aes_key = None
```

```
    self.sending_tcp_data_lock = threading.Lock
```

```
    self.initiate_rsa_protocol
```

```
    def get_aes_key(self
```

```
        return self.aes_key
```

```
    def receive_by_size(self, size, buffer_size=16384
```

```

        ()received_data = bytearray

        :while len(received_data) < size
        (remaining_size = size - len(received_data
        :try
        ((chunk = self.server.recv(min(buffer_size, remaining_size
        :except socket.error as e
        Handle socket errors, e.g., connection reset #
        ("{print(f"Socket error: {e
        return None

        :if not chunk
        Connection closed #
        return None

        (received_data.extend(chunk

        (return bytes(received_data

        :(def send_str(self, data
        :try
        Convert the length of the data to a string #
        ()self.sending_tcp_data_lock.acquire
        ('encoded_data = data.encode('utf-8
        encoded_encrypted_data = encrypt_with_aes(self.aes_key,
        (encoded_data

        Use the size of encoded_encrypted_data #
        ((size_str = str(len(encoded_encrypted_data

```

```

        Padding adjustment #
(number_of_zero = self.original_len - len(size_str
    size = ("0" * number_of_zero) + size_str
        Send the size as a string #
        (('self.server.send(size.encode('utf-8

        Send the actual data #
(self.server.send(encoded_encrypted_data
        :except socket.error as e
            (print(e
                :finally

        Release the lock #
        ()self.sending_tcp_data_lock.release

        :(def send_bytes(self, data
            :try

            ()self.sending_tcp_data_lock.acquire
            :if self.aes_key is None
                ((size_str = str(len(data
                    ((size = str(self.size + int(size_str
(number_of_zero = self.original_len - len(size
    size = ("0" * number_of_zero) + size
        Send the size as a string #
        (('self.server.send(size.encode('utf-8
            Send the actual data as bytes #
            (self.server.send(data
                :else

```

```

        (encrypted_data = encrypt_with_aes(self.aes_key, data
            ((size_str = str(len(encrypted_data
                ((size = str(self.size + int(size_str
                    (number_of_zero = self.original_len - len(size
                        size = ("0" * number_of_zero) + size
                            Send the size as a string #
                                (('self.server.send(size.encode('utf-8
                                    Send the actual data as bytes #
                                        (self.server.send(encrypted_data
                                            :except socket.error as e
                                                (print(e
                                                    :finally
                                                        Release the lock #
                                                            ()self.sending_tcp_data_lock.release

:(def send_message_dict_tcp(self, message_dict
    :try
        (pickled_data = pickle.dumps(message_dict
            (self.send_bytes(pickled_data
                :except Exception as e
                    (print(e

:(def send_played_song_bytes(self, song_bytes, title
message = {"message_type": "playlist_current_song_bytes", "audio_bytes":
    song_bytes, "title": title
    {
        (self.send_message_dict_tcp(message

:(def send_searched_song_info(self, searched_song_dict

```

```

message = {"message_type": "searched_song_result",
           "searched_song_dict": searched_song_dict
           }

(self.send_message_dict_tcp(message

:(def send_settings_dict(self, settings_dict
message = {"message_type": "settings_dict", "settings_dict": settings_dict
           }

(self.send_message_dict_tcp(message

:(def send_messages_list(self, messages_list
(json_messages_list = json.dumps(messages_list
message = {"message_type": "messages_list", "messages_list":
           json_messages_list
           }

(self.send_message_dict_tcp(message

:(def send_addition_messages_list(self, addition_messages_list
(json_messages_list = json.dumps(addition_messages_list
message = {"message_type": "message_list_addition",
           "message_list_addition": json_messages_list
           }

(self.send_message_dict_tcp(message

:(def send_new_message_content(self, chat, message_dict
, "message = {"message_type": "new_message
, chat_name": chat"
(message_dict": json.dumps(message_dict"
           }

```

```

(self.send_message_dict_tcp(message

                                :(def send_requests_list(self, list
                                (json_requests_list = json.dumps(list
message = {"message_type": "requests_list", "requests_list":
                                json_requests_list
                                {
(self.send_message_dict_tcp(message

                                :(def sent_code_to_mail(self
                                :try
                                Convert the length of the data to a string #
"message = {"message_type": "code", "action": "sent", "sent_to": "email
                                {
                                (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

                                :(def sent_friend_request_status(self, status
                                :try
                                Convert the length of the data to a string #
message = {"message_type": "friend_request", "friend_request_status":
                                {status
                                (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

                                :(def send_vc_data(self, vc_data, speaker
                                :try

```

```

        (compressed_vc_data = zlib.compress(vc_data
message = {"message_type": "vc_data", "compressed_vc_data":
        compressed_vc_data, "speaker": speaker
        {
            (self.send_message_dict_tcp(message
                :except Exception as e
                ("{print(f"error in send vc data is: {e

def send_share_screen_data(self, share_screen_data, speaker,
                            :(shape_of_frame_bytes
                            :try
(compressed_share_screen_data = zlib.compress(share_screen_data
    message = {"message_type": "share_screen_data",
                :""compressed_share_screen_data
compressed_share_screen_data, "speaker": speaker, "frame_shape":
                            shape_of_frame_bytes
                            {
                                (self.send_message_dict_tcp(message
                                    :except Exception as e
                                    ("{print(f"error in send share screen data is: {e

def send_share_camera_data(self, share_camera_data, speaker,
                            :(shape_of_frame_bytes
                            :try
(compressed_share_screen_data = zlib.compress(share_camera_data
    , "message = {"message_type": "share_camera_data
                    compressed_share_camera_data"."
                    ,compressed_share_screen_data
speaker": speaker, "frame_shape": shape_of_frame_bytes"
                            {

```

```

        (self.send_message_dict_tcp(message
                                     :except Exception as e
        ("{print(f"error in send camera data is: {e

:(def send_to_client_he_has_all_of_the_messages(self
, "message = {"message_type": "messages_status
               "messages_status": "up_to_data"
               {
               (self.send_message_dict_tcp(message

               :(def add_new_chat(self, chat_to_add
, "message = {"message_type": "add_chat
               chat_to_add": chat_to_add"
               {
               (self.send_message_dict_tcp(message

               :(def send_new_group(self, group_dict
, "message = {"message_type": "new_group_dict
               (group_dict": json.dumps(group_dict"
               {
               (self.send_message_dict_tcp(message

               :(def update_group(self, group_dict
, "message = {"message_type": "update_group_dict
               (group_dict": json.dumps(group_dict"
               {
               (self.send_message_dict_tcp(message

```



```

        : (def send_friends_list(self, friends_list
(json_friends_list = json.dumps(friends_list
, "message = {"message_type": "friends_list
friends_list": json_friends_list"
{
(self.send_message_dict_tcp(message

: (def playlist_songs_list(self, songs_dicts_list
(pickled_songs_list = pickle.dumps(songs_dicts_list
, "message = {"message_type": "playlist_songs
playlist_songs_list": pickled_songs_list"
{
(self.send_message_dict_tcp(message

: (def send_blocked_list(self, blocked_list
(json_blocked_list = json.dumps(blocked_list
, "message = {"message_type": "blocked_list
blocked_list": json_blocked_list"
{
(self.send_message_dict_tcp(message

: (def send_online_users_list(self, online_users_list
(json_online_users_list = json.dumps(online_users_list
, "message = {"message_type": "online_users_list
online_users_list": json_online_users_list"
{
(self.send_message_dict_tcp(message

```

```

:(def send_user_groups_list(self, group_list
  (json_group_list = json.dumps(group_list
, "message = {"message_type": "groups_list
  groups_list": json_group_list"
                                {
  (self.send_message_dict_tcp(message

:(def send_user_chats_list(self, chats_list
  (json_chats_list = json.dumps(chats_list
, "message = {"message_type": "chats_list
  chats_list": json_chats_list"
                                {
  (self.send_message_dict_tcp(message

:(def send_user_that_calling(self, user_that_is_calling
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
  {action": "calling", "user_that_called": user_that_is_calling"
  (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_user_that_call_ended(self
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
  {"action": "ended"
  (self.send_message_dict_tcp(message
                                :except socket.error as e

```

```

                                (print(e

                                :(def send_user_that_call_accepted(self
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
                                {"action": "accepted"
                                (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

                                :(def send_user_call_timeout(self
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
                                {"action": "timeout"
                                (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

                                :(def send_call_dict(self, call_dict
, "message = {"message_type": "call", "call_action_type": "call_dictionary
                                {action": "dict", "dict": call_dict"
                                (self.send_message_dict_tcp(message

                                :(def send_call_list_of_dicts(self, call_dicts_list
                                (json_call_dicts_list = json.dumps(call_dicts_list
, "message = {"message_type": "call", "call_action_type": "call_dictionary
                                {action": "list_call_dicts", "list_call_dicts": json_call_dicts_list"
                                (self.send_message_dict_tcp(message

```

```

:(def send_profile_list_of_dicts(self, profile_dicts_list
message = {"message_type": "profile_dicts_list", "profile_dicts_list":
           {profile_dicts_list
           (self.send_message_dict_tcp(message

:(def send_profile_dict_of_user(self, profile_dict, user
(json_profile_dict = json.dumps(profile_dict
message = {"message_type": "updated_profile_dict", "profile_dict":
           {json_profile_dict, "username": user
           (self.send_message_dict_tcp(message

:(def remove_call_to_user_of_id(self, call_id
:try
,"message = {"message_type": "call", "call_action_type": "update_calls
           {action": "remove_id", "removed_id": call_id"
           (self.send_message_dict_tcp(message
           :except socket.error as e
           (print(e

:(def send_stream_of_user_closed(self, user
:try
,"message = {"message_type": "call", "call_action_type": "in_call_action
           {action": "stream_stopped", "user_that_stopped": user"
           (self.send_message_dict_tcp(message
           :except socket.error as e
           (print(e

:(def send_confirm_login(self

```

```

:try
{"message" = {"message_type": "login", "login_status": "confirm
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_invalid_login(self
:try
{"message" = {"message_type": "login", "login_status": "invalid
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_already_logged_in(self
:try
{"message" = {"message_type": "login", "login_status": "already_logged_in
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_2fa_on(self
:try
{"message" = {"message_type": "login", "login_status": "2fa
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_sign_up_confirm(self

```

```

:try
{"message": {"message_type": "sign_up", "sign_up_status": "confirm"
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_sign_up_invalid(self
:try
{"message": {"message_type": "sign_up", "sign_up_status": "invalid"
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_sign_up_code_invalid(self
:try
message = {"message_type": "sign_up", "action": "code", 'code_status':
{"invalid"
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_sign_up_code_valid(self
:try
message = {"message_type": "sign_up", "action": "code", 'code_status':
{"valid"
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

```

```

:(def send_all_data_received(self
                                :try
message = {"message_type": "data", "action": "receive", 'receive_status':
                                                {"done"
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def send_security_token_to_client(self, security_token
                                :try
message = {"message_type": "security_token", "security_token":
                                                {security_token
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def send_security_token_valid(self
                                :try
{"message = {"message_type": "security_token", "security_status": "valid
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def send_security_token_invalid(self
                                :try
message = {"message_type": "security_token", "security_status":
                                                {"invalid
(self.send_message_dict_tcp(message
                                :except socket.error as e

```

```

                                (print(e

:(def send_forget_password_info_valid(self
                                :try
message = {"message_type": "forget_password",
          {"forget_password_status": "valid
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_forget_password_info_invalid(self
                                :try
message = {"message_type": "forget_password",
          {"forget_password_status": "invalid
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_forget_password_code_valid(self
                                :try
message = {"message_type": "forget_password", "action": "code",
          {"code_status": "valid
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_2fa_code_valid(self
                                :try
message = {"message_type": "2fa", "action": "code", "code_status":
          {"valid

```



```

        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def send_2fa_code_invalid(self
        :try
message = {"message_type": "2fa", "action": "code", "code_status":
        {""invalid
        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def send_forget_password_code_invalid(self
        :try
message = {"message_type": "forget_password", "action": "code",
        {"code_status": "invalid
        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def timeout_receive(self
        :try
{"message = {"message_type": "timeout_receive
        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def recv_str(self
        :try

```

```

Receive the size as binary data and convert it to an integer #
('size_str = self.server.recv(self.original_len).decode('utf-8

Convert the size string to an integer #
(size = int(size_str

Receive the actual data based on the size #
(data = self.receive_by_size(size
:try
    encrypted_data = data
    :if data is None
        ("print("Received data is None
            return None
(data = decrypt_with_aes(self.aes_key, encrypted_data
    (return pickle.loads(data
        :except Exception as e
If decoding as UTF-8 fails, treat it as binary data #
    ("{print(f"error in receiving data: {e
        return data

:except (socket.error, ValueError) as e
    ("{self.logger.error(f"Error: {e
    ("...self.logger.info("Clearing socket buffer
Clear the socket buffer by receiving and discarding remaining data #
        return None

:(def recv_bytes(self
:try

```

```

Receive the size as binary data and convert it to an integer #
('size_str = self.server.recv(self.original_len).decode('utf-8

Convert the size string to an integer #
(size = int(size_str

Receive the actual data based on the size #
(data = self.receive_by_size(size
return data

except (socket.error, ValueError) as e
("{print(f"Error: {e
return None # Return None in case of an error

:(def return_socket(self
return self.server

:(def close(self
:try
()self.server.close
except socket.error as e
(print(e

:(def initiate_rsa_protocol(self
)server_public_key, server_private_key = generate_rsa_key_pair

send the server_public_key to the client #
)serialized_server_public_key = server_public_key.public_bytes

```

```

        ,encoding=serialization.Encoding.PEM
        format=serialization.PublicFormat.SubjectPublicKeyInfo
    )

    'public_key_byte_sequence = br"\server:public-key
(self.send_bytes(public_key_byte_sequence + serialized_server_public_key

    'symmetric_key_byte_sequence = br"\server:symmetric-key

    ()received_encrypted_symmetric_key_bytes = self.recv_bytes
        if
received_encrypted_symmetric_key_bytes.startswith(symmetric_key_byte_sequ
        :(ence

        received_encrypted_symmetric_key_bytes =
[: (received_encrypted_symmetric_key_bytes[len(symmetric_key_byte_sequence
        :else

        ("self.logger.critical("did not expect this kind of message
        return

        :if received_encrypted_symmetric_key_bytes is not None
        decrypted_symmetric_key = decrypt_with_rsa(server_private_key,
            (received_encrypted_symmetric_key_bytes
            ()aes_key = generate_aes_key

        self.logger.info(f"Started to communicate with client
        ("{{self.client_tcp_socket_address}}, with AES key {aes_key
        :try

        encrypted_aes_key = encrypt_with_aes(decrypted_symmetric_key,
            (aes_key

        (self.send_bytes(symmetric_key_byte_sequence + encrypted_aes_key

        self.aes_key = aes_key

        :except Exception as e

```

```

                                (print(e
                                :else
                                ".print("Error receiving the symmetric key

                                from datetime import datetime
                                import time
                                import uuid
                                import database_func
                                import copy
                                import threading
                                import base64
                                import pickle
from server_net import decrypt_with_aes, encrypt_with_aes, slice_up_data
                                import zlib
                                import socket
                                import datetime as date
                                import logging

                                :(def get_id_from_format(str_format
                                [temp = str_format.split("(")[1
                                [temp = temp.split(")")[0
                                (return int(temp

                                :(def find_common_elements(list1, list2
Use set intersection to find common elements #

```

```

((common_elements = list(set(list1) & set(list2
                                return common_elements

:(def create_profile_pic_dict(username, image_bytes_encoded
    :(if isinstance(image_bytes_encoded, bytes
        image_bytes_encoded =
        ('base64.b64encode(image_bytes_encoded).decode('utf-8

    } = current_dict
    ,username": username"
    ,encoded_image_bytes": image_bytes_encoded"
    {
    return current_dict

:(def remove_duplicates(input_list
    [] = unique_elements
    :for item in input_list
    :if item not in unique_elements
    (unique_elements.append(item
    return unique_elements

:(def relevant_users_for_user(user
    (user_friends = database_func.get_user_friends(user
    (user_groups = database_func.get_user_groups(user
    [] = total_groups_participants
    :for group in user_groups

```

```

total_groups_participants = total_groups_participants +
    ("group.get("group_members
    (total_groups_participants.append(user
total_needed_profile_names = remove_duplicates(total_groups_participants +
    (user_friends
    return total_needed_profile_names

```

```

:(def get_list_of_needed_profile_dict(user
    [] = list_needed_profile_dicts
    (total_needed_profile_names = relevant_users_for_user(user
    :for name in total_needed_profile_names
    (profile_pic_bytes = database_func.get_profile_pic_by_name(name
    (current_dict = create_profile_pic_dict(name, profile_pic_bytes
    (list_needed_profile_dicts.append(current_dict
    return list_needed_profile_dicts

```

```

:class Call
:(def __init__(self, parent, participants, nets, group_id=None
    (__self.logger = logging.getLogger(__name
    self.nets_dict = nets
    self.parent = parent
    :if group_id is not None
    self.is_group_call = True
    self.group_id = group_id
    :else
    self.is_group_call = False
    self.participants = participants

```

```

        existed_ring_id_associated_with_call =
(self.parent.get_ring_id_by_possible_ringers(self.participants
        :if existed_ring_id_associated_with_call is not None
        self.call_id = existed_ring_id_associated_with_call
        :else
        self.call_id = str(uuid.uuid4()) # Generates a random UUID
self.logger.info(f"Call of id ({self.call_id}) was created. Users in call are
        ({{self.participants
        {} = self.call_nets
        ()self.initiated_time = datetime.now
        ()self.gets_call_nets_from_dict
        [] = self.muted
        [] = self.deafened
        [] = self.video_streams_list
        ()self.send_call_object_to_clients
        ()self.send_to_everyone_call_accepted
        if using thread here #
        self.data_collection = [] # List of tuples containing user and vc_data
self.stop_thread = threading.Event() # Event for signaling the thread to stop
        (self.thread = threading.Thread(target=self.process_vc_data
        ()self.thread.start

:(def create_video_stream_of_user(self, user, type_of_stream
        :if self.is_group_call
video_stream = VideoStream(self.parent, user, self, type_of_stream,
        (self.group_id
        :else
video_stream = VideoStream(self.parent, user, self, type_of_stream,
        (None
        (self.video_streams_list.append(video_stream

```



```

        ()self.send_call_object_to_clients

:(def close_video_stream_by_user(self, user, type_of_stream
    :for stream in self.video_streams_list
:if stream.streamer == user and stream.stream_type == type_of_stream
        ()stream.end_stream
    (self.video_streams_list.remove(stream
        ()self.send_call_object_to_clients

:(def get_call_group_members(self
(temp_list = database_func.get_group_members(self.group_id
    return temp_list

:(def get_call_dict(self
    } = call_data
,is_group_call": self.is_group_call"
    ,call_id": self.call_id"
    ,participants": self.participants"
    ,muted": self.muted"
    ,deafened": self.deafened"
    ,()screen_streamers": self.get_all_video_screen_streamers"
    ,()camera_streamers": self.get_all_video_camera_streamers"
    ,group_id": self.group_id if self.is_group_call else None"
    Add more attributes as needed #
    {
    return call_data

:(def get_all_video_screen_streamers(self

```

```

        [] = list_names
        :for video_stream in self.video_streams_list
        :if video_stream.stream_type == "ScreenStream
        (list_names.append(video_stream.streamer
        return list_names

        : (def get_all_video_camera_streamers(self
        [] = list_names
        :for video_stream in self.video_streams_list
        :if video_stream.stream_type == "CameraStream
        (list_names.append(video_stream.streamer
        return list_names

        : (def send_call_object_to_clients(self
        Extract relevant attributes to send #
        ()call_data = self.get_call_dict
        : ()for name, net in self.call_nets.items
        :if net is not None
        (net.send_call_dict(call_data

        : (def call_ending_protocol(self
        ("{self.logger.debug(f"call participants: {self.participants
        : ()for name, net in self.call_nets.items
        :if net is not None
        ()net.send_user_that_call_ended
        (net.remove_call_to_user_of_id(self.call_id
        ()self.close_all_video_streams
        ("self.logger.info(f"Call of id {self.call_id} ended

```

```

call_time = datetime.now() - self.initiated_time
("{self.logger.info(f"Call was up for {call_time
(self.parent.cancel_ring_by_id(self.call_id
        ()self.stop_processing

:(def close_all_video_streams(self
:for stream in self.video_streams_list
        ()stream.end_stream

:(def send_to_everyone_call_accepted(self
:()for name, net in self.call_nets.items
:if net is not None and name in self.participants
        ()net.send_user_that_call_accepted

:(def is_user_in_a_call(self, user
        :if user in self.participants
            return True
        :else
            return False

:(def gets_call_nets_from_dict(self
        :if not self.is_group_call
            temp_list = self.participants
        :else
            (temp_list = database_func.get_group_members(self.group_id
Create a dictionary with names and corresponding nets for names in #
            temp_list

self.call_nets = {name: self.parent.nets_dict.get(name) for name in
                    {temp_list

```

```

                                : (def update_call_nets(self
self.nets_dict = self.parent.nets_dict
                                ()self.gets_call_nets_from_dict

                                : (def remove_user_from_call(self, user
                                (self.participants.remove(user
                                ()self.gets_call_nets_from_dict
                                :for stream in self.video_streams_list
                                :if stream.streamer == user
                                ()stream.end_stream
                                :if user in stream.spectators
                                (stream.remove_spectator(user
                                ()self.send_call_object_to_clients

                                ("{self.logger.info(f"{user} left call by id {self.call_id

                                : (def add_user_to_call(self, user
                                (self.participants.append(user
                                ()self.gets_call_nets_from_dict
                                :try
                                (net = self.parent.get_net_by_name(user
                                ()net.send_user_that_call_accepted
                                ("{self.logger.debug(f"Sent call:accepted to user {user}, with net {net
                                :except Exception as e
                                ("{self.logger.error(f"Error sending string: {e
                                ()self.send_call_object_to_clients
                                ("{self.logger.info(f"{user} joined call by id {self.call_id

```

```

                                : (def process_vc_data(self
                                : () while not self.stop_thread.is_set
                                : if self.data_collection
                                (user, vc_data = self.data_collection.pop(0
                                (self.send_vc_data_to_everyone_but_user(vc_data, user
                                : else
Sleep or perform other tasks if the data collection is empty #
                                (time.sleep(0.1

                                : (def stop_processing(self
                                : () self.stop_thread.set
                                self.thread.join() # Wait for the thread to finish

:(def adding_vc_data_to_user_call_thread_queue(self, user, vc_data
                                ((self.data_collection.append((user, vc_data

:(def send_vc_data_to_everyone_but_user(self, vc_data, user
                                : () for name, net in self.call_nets.items
if name != user and net is not None and name not in self.deafened and
                                : name in self.participants
                                (compressed_vc_data = zlib.compress(vc_data
self.parent.send_large_udp_data(user, name, compressed_vc_data,
                                ("vc_data", None
                                ("{"self.logger.debug(f"Sent voice chat data to {name #

                                : (def is_a_group_a_call(self
                                return self.is_group_call

```

```

:(def toggle_mute_for_user(self, user
    :if user in self.muted
        (self.muted.remove(user
            ("{self.logger.info(f'{user} unmuted himself in call of id {self.call_id
                :else
                    (self.muted.append(user
                        ("{self.logger.info(f'{user} muted himself in call of id {self.call_id
                            ()self.send_call_object_to_clients

:(def toggle_deafen_for_user(self, user
    :if user in self.deafened
        (self.deafened.remove(user
            ("{self.logger.info(f'{user} undeafened himself in call of id {self.call_id
                :else
                    (self.deafened.append(user
                        ("{self.logger.info(f'{user} deafened himself in call of id {self.call_id
                            ()self.send_call_object_to_clients

:(def add_spectator_for_stream(self, spectator, streamer, stream_type
    :for stream in self.video_streams_list
:if stream.streamer == streamer and stream.stream_type == stream_type
    (stream.add_spectator(spectator
self.logger.info(f'{spectator} started watching stream of id {self.call_id}
    ("{and type {stream.stream_type
        return
    ("self.logger.error(f'couldn't add spectator to stream

:(def remove_spectator_for_stream(self, spectator
    :for stream in self.video_streams_list

```

```

        :if spectator in stream.spectators
        (stream.remove_spectator(spectator

class Ring
: (def __init__(self, Parent, ringer, nets, ringing_to=None, group_id=None
    (__self.logger = logging.getLogger(__name
        self.parent = Parent
        self.ring_time = 25
        self.nets_dict = nets
        self.ringer = ringer
        self.ring_id = str(uuid.uuid4()) # Generates a random UUID
        :if group_id is not None
        ("self.logger.info(f"Ring of id ({self.ring_id}) is a ring from type Group
            self.is_group_ring = True
        (group_members = database_func.get_group_members(group_id
            self.ringing_to = group_members
            (self.ringing_to.remove(ringer
                self.group_id = group_id
            :else
                self.is_group_ring = False
                self.ringing_to = ringing_to
            ()self.initiated_time = datetime.now
                {} = self.ringers_nets
                [] = self.ringed_to
            ()self.gets_ringing_nets_from_dict
                [] = self.already_ringed_to
            ()self.ring_to_everyone_online

```

```

        self.ring_thread_flag = True
        [] = self.accepted_rings
        [] = self.rejected_rings
        self.is_ringer_stopped_call = False
        threading.Thread(target=self.process_call_and_send_response,
                        ()args=()).start

        : (def rejected_ring(self, user
        (self.rejected_rings.append(user
        ("self.logger.info(f'{user} declined call

        : (def accepted_ring(self, user
        (self.accepted_rings.append(user

        : (def process_call_and_send_response(self
        time_counter = 0
        : while self.ring_thread_flag
        (time.sleep(0.1
        time_counter += 0.1
        : if time_counter >= self.ring_time
        self.ring_thread_flag = False
        if len(self.ringing_to) == (len(self.accepted_rings) +
        : ((len(self.rejected_rings
        self.ring_thread_flag = False
        self.logger.info(f"Ring of id {self.ring_id} was stopped due to everyone
        ("answering or rejecting ring

        : if not self.is_ringer_stopped_call
        ()self.stop_ring_for_unanswered_users
        : if len(self.accepted_rings) == 0

```



```

        ()self.stop_ring_for_ringer
self.logger.info(f"Ring of id {self.ring_id} was stopped due to no
                    ("answering the call
                    (self.parent.remove_ring(self

:(def stop_ring_for_unanswered_users(self
:()for name, net in self.ringers_nets.items
    :if net is not None
:if name not in self.accepted_rings and name not in self.rejected_rings
    ()net.send_user_call_timeout

:(def stop_ring_for_ringer(self
(ringers_net = self.parent.get_net_by_name(self.ringer
    ()ringers_net.send_user_call_timeout

:(def cancel_ring_for_all(self
:()for name, net in self.ringers_nets.items
    :if net is not None
    ()net.send_user_call_timeout
    ()self.stop_ring_for_ringer
    self.is_ringer_stopped_call = True
    self.ring_thread_flag = False
    (self.parent.remove_ring(self
self.logger.info(f"Ringer of ring id {self.ring_id} stopped ringing, {self.ringer}
                    ("...got frustrated

:(def gets_ringing_nets_from_dict(self
    temp_list = self.ringing_to

```

```

        Create a dictionary with names and corresponding nets for names in #
                                                    temp_list

        {self.ringers_nets = {name: self.nets_dict.get(name) for name in temp_list

                                                    :(def ring_to_everyone_online(self
                                                    :if self.is_group_ring
        (group_name = database_func.get_group_name_by_id(self.group_id
        "{format = f"({self.group_id}){group_name})({self.ringer
        :())for name, net in self.ringers_nets.items
        :if net is not None
        (net.send_user_that_calling(format
        (self.ringed_to.append(name
        ("{self.logger.info(f"Sent ring of id {self.ring_id} to {name
        (self.already_ringed_to.append(name
                                                    :else
        self.logger.info(f"Created 1 on 1 ring (id={self.ring_id}) [{self.ringing_to[0]},
        ("[{ringer is:{self.ringer
        :())for name, net in self.ringers_nets.items
        :if net is not None
        (net.send_user_that_calling(self.ringer
        (self.ringed_to.append(name
        ("{self.logger.info(f"Sent ring of id {self.ring_id} to {name
        (self.already_ringed_to.append(name

                                                    :(def update_ring_nets(self, nets
                                                    self.nets_dict = nets
                                                    ()self.gets_ringing_nets_from_dict
                                                    ()self.ring_to_users_who_didnt_get_a_ring

```

```

        : (def ring_to_users_who_didnt_get_a_ring(self
            : if self.is_group_ring
(group_name = database_func.get_group_name_by_id(self.group_id
        "{format = f"({self.group_id}){group_name})({self.ringer
            : ()for name, net in self.ringers_nets.items
        : if name not in self.already_ringed_to and net is not None
            (net.send_user_that_calling(format
                (self.ringed_to.append(name
        ("{self.logger.info(f"Sent ring of id {self.ring_id} to {name
            (self.already_ringed_to.append(name
                                                    : else
            : ()for name, net in self.ringers_nets.items
        : if name not in self.already_ringed_to and net is not None
            (net.send_user_that_calling(self.ringer
                (self.ringed_to.append(name
        ("{self.logger.info(f"Sent ring of id {self.ring_id} to {name
            (self.already_ringed_to.append(name

: (def is_ring_by_ringer(self, ringer
    return self.ringer == ringer

```

```

: class ServerHandler
:     : (def __init__(self
        [] = self.calls
        [] = self.rings
        {} = self.nets_dict
        {} = self.udp_addresses_dict

```

```

        [] = self.online_users
        self.udp_socket = None
        (__self.logger = logging.getLogger(__name
        self.udp_socket = None
        [] = self.UDPClientHandler_list
        self.server_mtu = None

:(def update_message_for_users(self, users, message, chat_name=None
        :for user in users
            :(if self.is_user_online(user
                ("message_type = message.get("message_type
                    ("sender = message.get("sender
                        :if message_type == "add_message
                            ("content = message.get("content
                                ("type_of_message = message.get("type
                                    ("file_name = message.get("file_name
                                        ()time_now = date.datetime.now
                                    ('formatted_time = time_now.strftime('%Y-%m-%d %H:%M
                                        } = formatted_message
                                            ,content": content"
                                            ,sender_id": sender"
                                            ,timestamp": formatted_time"
                                        ,message_type": type_of_message"
                                            file_name": file_name"
                                            {
                            (n = self.get_net_by_name(user
                                (n.send_new_message_content(chat_name, formatted_message
                                ("{self.logger.info(f"send new message to {user} in chat {chat_name

```

```

        : (def is_user_online(self, user
            return user in self.online_users

        : (def add_udp_address(self, address, user
            self.udp_addresses_dict[user] = address

: (def send_bytes_udp(self, data, address_destination, sending_to_user
    : try
        Encrypt the data if encryption is enabled #
            aes_key = None
            : if sending_to_user is not None
                (aes_key = self.get_aes_key_by_username(sending_to_user
                    : if aes_key is not None
                        (encrypted_data = encrypt_with_aes(aes_key, data
                            data = encrypted_data
                        (self.udp_socket.sendto(data, address_destination
                            : except socket.error as e
                                ("{{print(f"error in sending udp {e} , data size = {len(data
                                    raise

: (def send_message_dict_udp(self, message_dict, address, username
    : try
        (pickled_data = pickle.dumps(message_dict
        (self.send_bytes_udp(pickled_data, address, username
            : except Exception as e
                (print(e

```

```

def send_large_udp_data(self, sender, sending_to, data, data_type,
                        :(shape_of_frame=None

    (address_to_send = self.udp_addresses_dict.get(sending_to
                                                    :if len(data) > self.server_mtu

    ((sliced_data = slice_up_data(data, int(self.server_mtu * 0.2
                                    :else

        [sliced_data = [data
                        index = 0

        :for data_slice in sliced_data
            :if index == 0
                is_first = True
            :else
                is_first = False
        :if index == len(sliced_data)-1
            is_last = True
        :else
            is_last = False

        ,message = {"message_type": data_type
                    ,is_first": is_first, "is_last": is_last"
sliced_data": data_slice, "shape_of_frame": shape_of_frame,"
                                                    {"speaker": sender

(self.send_message_dict_udp(message, address_to_send, sending_to
                                index += 1

:(def check_max_packet_size_udp(self, temp_address

    'data = b'a

destination_address = temp_address # Using Google's DNS server
                                    address

                                    :try

```

```

                                :while True
(self.send_bytes_udp(data, destination_address, None
                                (data += (b'a' * 100
                                :except socket.error as e
                                ("self.logger.info(f"Network MTU is: {len(data)} - 10
                                self.server_mtu = len(data) - 10

                                :(def send_new_group_to_members(self, group_id
(group_members = database_func.get_group_members(group_id
                                (group_dict = database_func.get_group_by_id(group_id
                                :for member in group_members
                                (net = self.get_net_by_name(member
                                :if net is not None
                                (net.send_new_group(group_dict

                                :(def update_group_dict_for_members(self, group_id
(group_members = database_func.get_group_members(group_id
                                (group_dict = database_func.get_group_by_id(group_id
                                :for member in group_members
                                (net = self.get_net_by_name(member
                                :if net is not None
                                (net.update_group(group_dict

                                :(def send_to_user_needed_info(self, User
                                (net = self.get_net_by_name(User

                                (friends_list = database_func.get_user_friends(User
                                (net.send_friends_list(friends_list

```

```

        ({self.logger.info(f'Sent friend list ({friends_list}) to user {User
(friend_request_list = database_func.get_friend_requests(User
        (net.send_requests_list(friend_request_list
({self.logger.info(f'Sent requests list ({friend_request_list}) to user {User
        (user_groups_list = database_func.get_user_groups(User
        (net.send_user_groups_list(user_groups_list
        ({self.logger.info(f'Sent groups list to user {User
        (user_blocked_list = database_func.get_blocked_users(User
        (net.send_blocked_list(user_blocked_list
        ({self.logger.info(f'Sent blocked list to user {User
        (user_chats_list = database_func.get_user_chats(User
        (net.send_user_chats_list(user_chats_list
        ({self.logger.info(f'Sent Chats list to user {User
(online_friends = find_common_elements(self.online_users, friends_list
        (net.send_online_users_list(online_friends
        ({self.logger.info(f'Sent Online friends list to user {User
        (list_call_dicts = self.get_list_of_calls_for_user(User
        (net.send_call_list_of_dicts(list_call_dicts
        ({self.logger.info(f'Sent list of call dicts list to user {User

        (self.send_profile_list_of_dicts_to_user(User
        ({self.logger.info(f'Sent list of profile dicts list to user {User

        (songs_list = database_func.get_songs_by_owner(User
        (net.playlist_songs_list(songs_list
        ({self.logger.info(f'Sent list of songs dicts list to user {User

        (settings_dict = database_func.get_user_settings(User

```



```

                                :if settings_dict is not None
                                (net.send_settings_dict(settings_dict
                                ("{self.logger.info(f'Sent settings to user {User
                                :else
                                ("self.logger.error("couldn't find user's settings

                                ()net.send_all_data_received
                                ("{self.logger.info(f'All needed data sent to {User

def update_profiles_list_for_everyone_by_user(self, user,
                                              :(b64_encoded_profile_pic
                                              (relevant_users = relevant_users_for_user(user
                                              :for relevant_user in relevant_users
self.send_new_profile_of_user(relevant_user, b64_encoded_profile_pic,
                                              (user

def send_new_profile_of_user(self, user, b64_encoded_profile_pic,
                                :(user_of_profile
                                (net = self.get_net_by_name(user
                                :if net is not None
                                profile_dict = create_profile_pic_dict(user_of_profile,
                                (b64_encoded_profile_pic
                                (net.send_profile_dict_of_user(profile_dict, user_of_profile
self.logger.info(f'Sent list new profile dict of user {user_of_profile} to user
                                ("{{user

                                :(def send_profile_list_of_dicts_to_user(self, user
                                (net = self.get_net_by_name(user
                                :if net is not None
                                (list_profile_dicts = get_list_of_needed_profile_dict(user

```

```

        (net.send_profile_list_of_dicts(list_profile_dicts
    ("{self.logger.info(f'Sent list of profile dicts list to user {user

```

```

        : (def get_list_of_calls_for_user(self, user
            [] = list_of_calls_dicts
            :for call in self.calls
            :if call.is_group_call
        :()if user in call.get_call_group_members
        ()current_call_dict = call.get_call_dict
        (list_of_calls_dicts.append(current_call_dict
            return list_of_calls_dicts

```

```

        : (def update_online_list_for_users_friends(self, user
        (user_friends_list = database_func.get_user_friends(user
            :for friend in user_friends_list
            :if friend in self.online_users
        (friend_friends_list = database_func.get_user_friends(friend
            (friends_net = self.get_net_by_name(friend
        online_friends = find_common_elements(self.online_users,
            (friend_friends_list
        (friends_net.send_online_users_list(online_friends

```

```

        : (def user_online(self, user, net
        (self.online_users.append(user
            (self.add_net(user, net
            (self.send_to_user_needed_info(user
        (self.update_online_list_for_users_friends(user

```

```

        : (def user_offline(self, user

```

```

        (self.online_users.remove(user
        (self.remove_net_by_name(user
        ()self.update_nets_for_child_class
        (self.update_online_list_for_users_friends(user
        :(if self.is_user_in_a_call(user
        (self.remove_user_from_call(user

        :(def add_net(self, name, obj
        self.nets_dict[name] = obj
        ()self.update_nets_for_child_class

        :(def update_nets_for_child_class(self
        :for ring in self.rings
        (ring.update_ring_nets(self.nets_dict
        :for call in self.calls
        ()call.update_call_nets

        :(def get_net_by_name(self, name
        (return self.nets_dict.get(name, None

        :(def remove_net_by_name(self, name
        :if name in self.nets_dict
        [del self.nets_dict[name
        (".self.logger.info(f"Net '{name}' removed successfully
        :else
        (".self.logger.warning(f"Net '{name}' not found

        :(def add_call(self, call

```

```

                                :if len(call.participants) < 2
self.logger.error("A call must have at least 2 participants. Call was not
                                (".added
                                :else
                                (self.calls.append(call

                                :(def is_user_in_a_call(self, user
                                :for call in self.calls
                                :(if call.is_user_in_a_call(user
                                return True
                                return False

                                :(def remove_user_from_call(self, user
                                :if not self.calls
self.logger.warning("Tried to remove user from call. But there is No active
                                (".calls
                                return

                                :for call in self.calls
                                :if user in call.participants
                                :if len(call.participants) == 2
                                :if call.is_group_call
                                )self.logger.info
                                f"{user} ended call in group call of
                                ("{(id:{database_func.get_group_name_by_id(call.group_id
                                :else
                                (temp_list = copy.deepcopy(call.participants
                                (temp_list.remove(user
                                ("[{self.logger.info(f"{user} ended call with {temp_list[0

```

```

        ()call.call_ending_protocol
self.calls.remove(call) # Remove the call from the calls list
                                :else
        (call.remove_user_from_call(user
(users_net = self.get_net_by_name(user
        ()users_net.send_user_that_call_ended
("self.logger.info("Removed user from group call

:(def are_users_in_a_call(self, list_users
        :for call in self.calls
        :if call.participants == list_users
            return True
        return False

:(def get_call_members_with_user(self, user
        :for call in self.calls
        :if user in call.participants
return [participant for participant in call.participants if participant !=
                                                    [user
        return None

:(def create_call_and_add(self, group_id, participants
call = Call(parent=self, participants=participants, nets=self.nets_dict,
            (group_id=group_id
        (self.add_call(call

:(def add_ring(self, ring
        (self.rings.append(ring

```

```

:(def remove_ring(self, ring
                    :try
                    (self.rings.remove(ring
                    :except Exception as e
                    (self.logger.error(e

:(def create_ring(self, ringer, ringing_to
                  :(")")if ringing_to.startswith
                  (group_id = get_id_from_format(ringing_to
ring = Ring(Parent=self, ringer=ringer, nets=self.nets_dict,
            (group_id=group_id
            (self.add_ring(ring
                        :else
                        [temp_list = [ringing_to
ring = Ring(Parent=self, ringer=ringer, nets=self.nets_dict,
            (ringing_to=temp_list
            (self.add_ring(ring

:(def is_group_call_exist_by_id(self, group_id
                                :for call in self.calls
                                :if call.is_group_call
                                :if call.group_id == group_id
                                return True
                                return False

:(def send_vc_data_to_call(self, vc_data, User
                            :for call in self.calls
                            :(if call.is_user_in_a_call(User
(call.adding_vc_data_to_user_call_thread_queue(User, vc_data

```

```

(call.send_vc_data_to_everyone_but_user(vc_data, User #

def send_share_screen_data_to_call(self, share_screen_data,
                                   :(shape_bytes_of_frame, User, stream_type
                                   :for call in self.calls
                                   :(if call.is_user_in_a_call(User
                                   :for video_stream in call.video_streams_list
if video_stream.streamer == User and video_stream.stream_type ==
                                   :stream_type

video_stream.adding_share_screen_data_to_user_call_thread_queue(User,
                        (share_screen_data, shape_bytes_of_frame

:(def add_user_to_group_call_by_id(self, User, id
                                   :for call in self.calls
                                   :if call.is_a_group_a_call
                                   :if call.group_id == id
                                   (call.add_user_to_call(User

:(def reject_ring_by_ringer(self, ringer, User
                                   :for ring in self.rings
                                   :(if ring.is_ring_by_ringer(ringer
                                   (ring.rejected_ring(User
                                   break

:(def accept_ring_by_ringer(self, ringer, User
                                   :for ring in self.rings
                                   :(if ring.is_ring_by_ringer(ringer
                                   (ring.accepted_ring(User
                                   break

```

```

:(def cancel_ring_by_the_ringer(self, ringer
                                :for ring in self.rings
                                :if ring.is_ring_by_ringer(ringer
                                ()ring.cancel_ring_for_all

:(def mute_or_unmute_self_user(self, user
                                :for call in self.calls
                                :if call.is_user_in_a_call(user
                                (call.toggle_mute_for_user(user

:(def deafen_or_undeafen_self_user(self, user
                                :for call in self.calls
                                :if call.is_user_in_a_call(user
                                (call.toggle_deafen_for_user(user

:(def get_ring_id_by_possible_ringers(self, possible_ringers
                                :for ring in self.rings
                                :if ring.ringer in possible_ringers
                                return ring.ring_id
                                return None

:(def cancel_ring_by_id(self, ring_id
                                :for ring in self.rings
                                :if ring.ring_id == ring_id
                                ()ring.stop_ring_for_unanswered_users
                                (self.rings.remove(ring

```



```

:(def create_video_stream_for_user_call(self, user, type
                                     :for call in self.calls
                                     :if user in call.participants
    (call.create_video_stream_of_user(user, type

:(def close_video_stream_for_user_call(self, user, type
                                     :for call in self.calls
                                     :if user in call.participants
    (call.close_video_stream_by_user(user, type

:(def add_spectator_to_call_stream(self, spectator, streamer, stream_type
                                     :for call in self.calls
                                     :if (spectator and streamer) in call.participants
    (call.add_spectator_for_stream(spectator, streamer, stream_type

:(def remove_spectator_from_call_stream(self, spectator
                                     :for call in self.calls
                                     :if spectator in call.participants
    (call.remove_spectator_for_stream(spectator

:(def handle_udp_fragment(self, fragment, address
:for udp_handler_object in self.UDPClientHandler_list
:if udp_handler_object.udp_address == address
(udp_handler_object.handle_udp_message(fragment

def create_and_add_udp_handler_object(self, username, udp_address,
                                     :tcp_address
    (self.add_udp_address(udp_address, username

```

```

udp_handler_object = UDPClientHandler(udp_address, tcp_address, self,
                                      (username
                                      (self.UDPClientHandler_list.append(udp_handler_object

:
(def get_aes_key_by_username(self, username
  (user_net = self.get_net_by_name(username
    ()return user_net.get_aes_key

:
class VideoStream
def __init__(self, Comms_object, streamer, call_object, stream_type,
            :
            (group_id=None
            self.call_parent = call_object
            self.comms_parent = Comms_object
            (__self.logger = logging.getLogger(__name
            (())self.stream_id = str(uuid.uuid4
            self.streamer = streamer
            stream_type is either CameraStream or ScreenStream #
            self.stream_type = stream_type
            :
            if group_id
            self.is_group_stream = True
            :
            else
            self.is_group_stream = False
            [] = self.spectators
            self.data_collection = [] # List of tuples containing user and vc_data
self.stop_thread = threading.Event() # Event for signaling the thread to stop
(self.thread = threading.Thread(target=self.process_share_screen_data
  self.thread.info(f"Video Stream of id {self.stream_id} and type
                  ("{self.stream_type} was created

```

```

:(def remove_spectator(self, user
      (self.spectators.remove(user
self.logger.info(f"{user} stopped watching stream of {self.streamer} with id
                  ("{{self.stream_id
      :if len(self.spectators) == 0
      ()self.stop_processing
      ()self.data_collection.clear

      :(def add_spectator(self, user
      (self.spectators.append(user
self.logger.info(f"{user} started watching stream of {self.streamer} with id
                  ("{{self.stream_id
      :if len(self.spectators) == 1
self.stop_thread = threading.Event() # Event for signaling the thread to
                                     stop
(self.thread = threading.Thread(target=self.process_share_screen_data
                                ()self.thread.start
                                ("{{self.logger.info(f"Started stream thread of id {self.stream_id

:(def process_share_screen_data(self
      :()while not self.stop_thread.is_set
      :if self.data_collection
      user, share_screen, share_screen_frame_shape_bytes =
      (self.data_collection.pop(0
self.send_share_screen_data_to_everyone_but_user(share_screen,
      (user, share_screen_frame_shape_bytes
      :else
      Sleep or perform other tasks if the data collection is empty #
      (time.sleep(0.1

```

```

self.logger.info(f"stopped thread of video stream of id {self.stream_id} for
                                                         ("sssssure

                                                         :(def stop_processing(self
                                                         ()self.stop_thread.set
                                                         self.thread.join() # Wait for the thread to finish
                                                         ("{self.logger.info(f"stopped thread of video stream of id {self.stream_id

def send_share_screen_data_to_everyone_but_user(self, share_screen_data,
                                                         :(user, share_screen_frame_shape_bytes
                                                         :()for name, net in self.call_parent.call_nets.items
                                                         :if name != user and net is not None and name in self.spectators
                                                         :if self.stream_type == "CameraStream
net.send_share_camera_data(share_screen_data, user, #
                                                         (share_screen_frame_shape_bytes
                                                         compressed_share_screen_data =
                                                         (zlib.compress(share_screen_data
self.comms_parent.send_large_udp_data(user, name,
                                                         compressed_share_screen_data, "share_camera_data",
                                                         (share_screen_frame_shape_bytes
                                                         :else
net.send_share_screen_data(share_screen_data, user, #
                                                         (share_screen_frame_shape_bytes
                                                         compressed_share_screen_data =
                                                         (zlib.compress(share_screen_data
self.comms_parent.send_large_udp_data(user, name,
                                                         ,compressed_share_screen_data
                                                         share_screen_data",
                                                         (share_screen_frame_shape_bytes

                                                         :(def end_stream(self
                                                         :if len(self.spectators) > 0

```

```

        )self.stop_processing

("self.logger.info(f"Video Stream of id {self.stream_id} ended

def adding_share_screen_data_to_user_call_thread_queue(self, user,
        :(share_screen_data, shape_bytes_of_frame
        :if len(self.spectators) > 0
        self.data_collection.append((user, share_screen_data,
        ((shape_bytes_of_frame

        :class UDPClientHandler

def __init__(self, udp_address, tcp_address, ServerHandler_object,
        :(client_username

        (__self.logger = logging.getLogger(__name
        self.udp_address = udp_address
        self.tcp_address = tcp_address
        self.ServerHandler_object = ServerHandler_object
        self.client_username = client_username

        self.logger.info(f"create udp client handler of udp address
        {self.udp_address} and tcp address of {self.tcp_address} of username
        ("{{self.client_username

        self.aes_key =
(self.ServerHandler_object.get_aes_key_by_username(self.client_username

        self.lost_packets = 0
        self.gotten_packets = 0

        [] = self.vc_data
        [] = self.share_screen_data
        [] = self.share_camera_data

        :(def decrypt_data(self, data

```

```

        (return decrypt_with_aes(self.aes_key, data

        :(def handle_udp_message(self, fragment

            :try

            (pickled_data = self.decrypt_data(fragment

                (data = pickle.loads(pickled_data

                ("message_type = data.get("message_type

                    ("is_first = data.get("is_first

                    ("is_last = data.get("is_last

                    ("sliced_data = data.get("sliced_data

                ("shape_of_frame = data.get("shape_of_frame

                    :if message_type == "vc_data

self.handle_data_fragment(self.vc_data, sliced_data, is_first, is_last,

                                                                    (""vc_data

                    :elif message_type == "share_screen_data

self.handle_data_fragment(self.share_screen_data, sliced_data,

                                                                    ,is_first, is_last, "ScreenStream

                                                                    (shape_of_frame

                    :elif message_type == "share_camera_data

self.handle_data_fragment(self.share_camera_data, sliced_data,

                                                                    ,is_first, is_last, "CameraStream

                                                                    (shape_of_frame

                                                                    self.gotten_packets += 1

                                                                    :except

                                                                    self.lost_packets += 1

                                                                    self.gotten_packets += 1

def handle_data_fragment(self, data_list, sliced_data, is_first, is_last,

                                                                    :(data_type, shape_of_frame=None

                                                                    :if is_first

```

```

        ()data_list.clear
        (data_list.append(sliced_data

                                :if is_last
                                (full_data = b".join(data_list
                                (decompressed_data = zlib.decompress(full_data
                                :if data_type == "vc_data
self.ServerHandler_object.send_vc_data_to_call(decompressed_data,
                                                (self.client_username
                                :elif data_type == "ScreenStream

self.ServerHandler_object.send_share_screen_data_to_call(decompressed_data
                                                         ,, shape_of_frame

(self.client_username, data_type

                                :elif data_type == "CameraStream

self.ServerHandler_object.send_share_screen_data_to_call(decompressed_data
                                                         ,, shape_of_frame

(self.client_username, data_type

                                ()data_list.clear

```

```

import sys

* from PyQt5.QtWidgets import
from PyQt5.QtGui import QPixmap, QIntValidator, QIcon, QImage
\ ,from PyQt5.QtCore import QSize, QPoint, QTimer, QMetaObject, pyqtSignal
QSettings, Qt, QUrl, QTime
from PyQt5.QtMultimedia import QMediaPlayer, QMediaContent
from client_net import ClientNet

```

```

        from social_page_widgets import FriendsBox
        from settings_page_widgets import SettingsBox
    from messages_page_widgets import ChatBox, play_mp3_from_bytes,
        make_q_object_clear

    \ ,from chat_file import VideoPlayer, PlaylistWidget, get_camera_names
    \ ,make_circular_image, find_output_device_index, find_input_device_index
        get_default_output_device_name, get_default_input_device_name

        import pyaudio
        import random
        import json
        import threading
        import time
        import zlib
        import base64
        import datetime
        import pickle
        import cv2
        import numpy as np
        import pyautogui
        import re

        from PIL import ImageGrab
    from PyQt5.QtWidgets import QSpacerItem, QSizePolicy

    email_providers = ["gmail", "outlook", "yahoo", "aol", "protonmail", "zoho", "mail",
        ,""fastmail", "gmx", "yandex
        ,"mail.ru"

        ["tutanota", "icloud", "rackspace", "mailchimp"

CAMERA_FPS = 60

```



```
FORMAT = pyaudio.paInt16
```

```
CHANNELS = 1
```

```
RATE = 44100
```

```
CHUNK = 4096
```

```
()p = pyaudio.PyAudio
```

```
SCREEN_FPS = 30
```

```
:(def find_difference(list1, list2
```

```
((return list(set(list1) - set(list2
```

```
:(def parse_group_caller_format(input_format
```

```
Define a regular expression pattern to capture the information #
```

```
('(\(+([()^]))\(+([()^])\(+pattern = re.compile(r'\((\d
```

```
Use the pattern to match the input_format #
```

```
(match = pattern.match(input_format
```

```
:if match
```

```
Extract the matched groups #
```

```
((group_id = int(match.group(1
```

```
()group_name = match.group(2).strip
```

```
()group_caller = match.group(3).strip
```

```
return group_id, group_name, group_caller
```

```
:else
```

```
Return None if no match is found #
```

```
return None
```

```
:(def duration_to_milliseconds(duration_str
  Split the duration string into minutes and seconds #
  ((':')minutes, seconds = map(int, duration_str.split

  Calculate the total duration in milliseconds #
  total_duration_ms = (minutes * 60 + seconds) * 1000

  return total_duration_ms
```

```
:(def save_token(token
  ("settings = QSettings("Connectify_App", "Connectify
    (settings.setValue("saved_security_token", token
```

```
:(def are_token_saved
  ("settings = QSettings("Connectify_App", "Connectify
  return settings.value("saved_security_token") is not None
```

```
:(def get_saved_token
  ("settings = QSettings("Connectify_App", "Connectify
    ("token = settings.value("saved_security_token
      return token
```

```

:()def delete_saved_token
("settings = QSettings("Connectify_App", "Connectify
("settings.remove("saved_security_token

```

```

:(def is_email_provider_in_list(email
    global email_providers
    :for i in email_providers
        :if i in email
            return True
        return False

```

```

def create_message_dict(content, sender_id, timestamp, message_type,
                        :(file_name
                            } = message_dict
                        ,content": content"
                        ,sender_id": sender_id"
                        ,(timestamp": str(timestamp"
                        ,message_type": message_type"
                        file_name": file_name"
                            {
                        return message_dict

```

```

:(def is_email_valid(email
    ()email = email.lower
    :if not is_email_provider_in_list(email) or "." not in email

```

```

        return False
    :if " " in email
        return False
    :elif '@' not in email
        return False
    :elif len(email) < 4
        return False
    ('@')parts = email.split
    :"" == [if len(parts) < 2 or parts[1
        return False
        return True

```

```

:(def is_string(variable
(return isinstance(variable, str

```

```

:(def audio_data_list_set_volume(datalist, volume
    """ Change value of list of audio chunks """
    sound_level = volume / 100.0

```

```

        [] = modified_datalist
    :for chunk_bytes in datalist
        Convert bytes to NumPy array of int16 #
(chunk_array = np.frombuffer(chunk_bytes, dtype=np.int16

```

```

        Scale audio data to adjust volume #
(scaled_chunk = (chunk_array * sound_level).astype(np.int16

```

```

        Convert back to bytes and append to modified_datalist #
        (())modified_datalist.append(scaled_chunk.tobytes

return modified_datalist

:(def set_camera_properties(cap
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640) # Set frame width to 1280
pixels
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480) # Set frame height to 720
pixels
cap.set(cv2.CAP_PROP_AUTOFOCUS, 0) # Disable autofocus
(cap.set(cv2.CAP_PROP_FOCUS, 0) # Set focus to minimum (if supported
(cap.set(cv2.CAP_PROP_ZOOM, 1) # Set zoom to minimum (if supported
cap.set(cv2.CAP_PROP_EXPOSURE, -7) # Set exposure to minimum (if
(supported

:(class SplashScreen(QWidget
()stop_loading_signal = pyqtSignal

:(def __init__(self, page_controller_object
()__super().__init
self.page_controller_object = page_controller_object
if self.page_controller_object.screen_width == 0 and
:self.page_controller_object.screen_height == 0
(screen = QDesktopWidget().screenGeometry
()self.page_controller_object.screen_height = screen.height
()self.page_controller_object.screen_width = screen.width

```

```

        ()self.init_ui

        :(def init_ui(self
            Set window properties #
            (self.stop_loading_signal.connect(self.close_page_open_main_page
                ('self.setWindowTitle('Splash Screen
                    """)self.setStyleSheet
                        } QWidget
/* background-color: #141c4b; /* Set your desired background color
        {
        } QLabel
        ;color: white
        /* font-size: 18px; /* Set your desired font size
        {
        (""
            Create logo label #
            (logo_label = QLabel(self
                )logo_label.setPixmap
QPixmap('discord_app_assets/connectify_icon.png')) # Replace with the
                actual path to your PNG file
            (logo_label.setAlignment(Qt.AlignCenter
                (logo_label.move(1690 // 2, 300

            Create loading label #
            (loading_label = QLabel('Loading...', self
            (loading_label.setAlignment(Qt.AlignCenter
            ('loading_label.setObjectName('loading_label
            (loading_label.move(1690 // 2 + 55, 460

```

```

Set up dot counter #
    self.dot_count = 0

Create timer to update loading dots #
    (self.loading_timer = QTimer(self
(self.loading_timer.timeout.connect(self.update_loading_dots

Set up elapsed time variable #
    self.elapsed_time = 0
    ()self.start_timer

Show the splash screen #
    ()self.show

:(def start_timer(self
self.loading_timer.start(400) # Update every 500 milliseconds

:(def close_page_open_main_page(self
    :try
    ()self.page_controller_object.change_to_main_page
    self.page_controller_object.is_logged_in = True
    ()self.hide
    :except Exception as e
        (print(e

:(def update_loading_dots(self
n = self.page_controller_object.n
    wait_time_sec = 5

```

```

        self.dot_count = (self.dot_count + 1) % 5
('loading_label = self.findChild(QLabel, 'loading_label
        (loading_label.setText('Loading' + '.' * self.dot_count
                                ()loading_label.adjustSize

                                Increment elapsed time #
self.elapsed_time += 500 # Timer interval is 500 milliseconds

if self.elapsed_time >= wait_time_sec * 1000: # 5 seconds
    Transition to the login page after 5 seconds #
:''' == if self.page_controller_object.main_page.username
        :()if not are_token_saved
        ()self.loading_timer.stop
()self.page_controller_object.change_to_login_page
        ()self.hide
        :else

        ()security_token = get_saved_token
        (n.send_security_token(security_token
        ()data = n.recv_str
        ("message_type = data.get("message_type
        :if message_type == "security_token
        ("server_answer = data.get("security_status
        :if server_answer == "valid
        ()data = n.recv_str
        ("message_type = data.get("message_type
        :if message_type == "login_action
        (":")parts = data.split
username, action_state = data.get("username"),
                                ("data.get("login_status

```



```

        : "if action_state == "valid
        ()self.loading_timer.stop
        ("print("logged in successfully
        :try
        ()self.hide
self.page_controller_object.main_page.username =
username
()self.page_controller_object.change_to_main_page
self.page_controller_object.is_logged_in = True

        ()self.page_controller_object.start_receive_thread_after_login
        ()self.hide
        :except Exception as e
        (print(e
        : "elif action_state == "invalid
        ("print("username already logged in
        : "elif server_answer == "invalid
        ("print("security token isn't valid
        ()self.loading_timer.stop
        ()self.page_controller_object.change_to_login_page
        ()self.hide

```

```

...class MainPage(QWidget): # main page doesnt know when chat is changed

```

```

        ()updated_chat_signal = pyqtSignal
        ()update_chat_page_without_messages_signal = pyqtSignal
        ()updated_social_page_signal = pyqtSignal
        ()getting_call_signal = pyqtSignal
        ()stop_sound_signal = pyqtSignal

```

```

        ()initiating_call_signal = pyqtSignal
        ()reset_call_var_signal = pyqtSignal
        ()new_message_play_audio_signal = pyqtSignal
        ()disconnect_signal = pyqtSignal
        ()stop_watching_stream_signal = pyqtSignal
        ()updated_settings_signal = pyqtSignal
        ()caching_circular_images_of_users_signal = pyqtSignal
        ()caching_circular_images_of_groups_signal = pyqtSignal
        (updating_profile_dict_signal = pyqtSignal(str, dict
            (update_group_lists_by_group = pyqtSignal(dict
(insert_messages_into_message_box_signal = pyqtSignal(list
        (insert_new_message_in_chat_signal = pyqtSignal(dict
(scroll_back_to_index_before_update_signal = pyqtSignal(int
        (insert_search_result_signal = pyqtSignal(dict
        (update_settings_from_dict_signal = pyqtSignal(dict
            ()update_message_box_signal = pyqtSignal
            ()close_call_threads_signal = pyqtSignal
            ()start_call_threads_signal = pyqtSignal
            ()insert_playlist_to_table_signal = pyqtSignal

    :(def __init__(self, Network, page_controller_object
        ()__super().__init
    self.page_controller_object = page_controller_object
        [] = self.vc_data_list

        self.vc_thread_flag = False
        self.send_vc_data_thread =
        ((=threading.Thread(target=self.thread_send_voice_chat_data, args

```

```

        self.vc_play_flag = False
        self.audio_data_lock = threading.Lock
        self.play_vc_data_thread =
        threading.Thread(target=self.thread_play_vc_data, args=

        self.vc_data_fragments_list
        self.share_screen_data_fragments_list
        self.share_camera_data_fragments_list
        self.listen_udp = True
        self.listen_udp_thread =
        threading.Thread(target=self.listen_udp_socket_thread, args=

        self.is_watching_screen = False
        self.watching_user
        self.watching_type = None

        self.is_screen_shared = False
        self.send_share_screen_thread =
        threading.Thread(target=self.thread_send_share_screen_data, args=

        self.is_camera_shared = False
        self.send_camera_data_thread =
        threading.Thread(target=self.thread_send_share_camera_data, args=

        self.regular_profile_image_path = "discord_app_assets/regular_profile.png

        self.blueish_background_color = "#141c4b
        self.blackish_background_color = "#000000
        self.reddish_background_color = "#5c1114
        self.grayish_background_color = "#363131

```

```

        "self.special_design_color = "#2b2d31

        "self.blueish_style_hover_color = "#2980b9
        "self.blackish_style_hover_color = "#FFFFFF
        "self.reddish_style_hover_color = "#7d1d21
        "self.grayish_style_hover_color = "#5c4a4b
        "self.special_design_hover_color = "#36373d

    self.hex_hover_colors = [self.reddish_style_hover_color,
                             ,self.blueish_style_hover_color
                             self.blackish_style_hover_color,
                             ,self.grayish_style_hover_color
                             [self.special_design_hover_color

    self.hex_colors = [self.reddish_background_color,
                       ,self.blueish_background_color, self.blackish_background_color
                       [self.grayish_background_color, self.special_design_color
    self.color_design_options = ["Red", "Blue", "Black and White", "Gray",
                                [""Connectify Special

                                } = self.color_design_mapping

                                ,[self.color_design_options[0]: self.hex_colors[0]
                                ,[self.color_design_options[1]: self.hex_colors[1]
                                ,[self.color_design_options[2]: self.hex_colors[2]
                                ,[self.color_design_options[3]: self.hex_colors[3]
                                [self.color_design_options[4]: self.hex_colors[4]

                                {

                                } = self.style_color_hover_mapping
                                ,[self.color_design_options[0]: self.hex_hover_colors[0]
                                ,[self.color_design_options[1]: self.hex_hover_colors[1]

```

```

,[self.color_design_options[2]: self.hex_hover_colors[2]
,[self.color_design_options[3]: self.hex_hover_colors[3]
,[self.color_design_options[4]: self.hex_hover_colors[4]
{
    } = self.special_keys_mapping
    ,"Qt.Key_Return: "Return
    ,"Qt.Key_Enter: "Enter
    ,"Qt.Key_Escape: "Escape
    ,"Qt.Key_Tab: "Tab
    ,"Qt.Key_Backspace: "Backspace
    ,"Qt.Key_Delete: "Delete
    ,"Qt.Key_Insert: "Insert
    ,"Qt.Key_Home: "Home
    ,"Qt.Key_End: "End
    ,"Qt.Key_PageUp: "Page Up
    ,"Qt.Key_PageDown: "Page Down
    ,"Qt.Key_Left: "Left Arrow
    ,"Qt.Key_Right: "Right Arrow
    ,"Qt.Key_Up: "Up Arrow
    ,"Qt.Key_Down: "Down Arrow
    ,"Qt.Key_F1: "F1
    ,"Qt.Key_F2: "F2
    ,"Qt.Key_F3: "F3
    ,"Qt.Key_F4: "F4
    ,"Qt.Key_F5: "F5
    ,"Qt.Key_F6: "F6
    ,"Qt.Key_F7: "F7
    ,"Qt.Key_F8: "F8

```

```

        , "Qt.Key_F9: "F9
        , "Qt.Key_F10: "F10
        , "Qt.Key_F11: "F11
        , "Qt.Key_F12: "F12
        , "Qt.Key_Shift: "Shift
        , "Qt.Key_Control: "Control
        , "Qt.Key_Alt: "Alt
        , "Qt.Key_Meta: "Meta/Windows
        , "Qt.Key_CapsLock: "Caps Lock
        , "Qt.Key_NumLock: "Num Lock
        , "Qt.Key_ScrollLock: "Scroll Lock
        Add more key mappings as needed #
    {
self.reversed_keys_mapping = {value: key for key, value in
    {()self.special_keys_mapping.items

self.standard_hover_color = "#2980b9"
self.background_color_hex = "#141c4b"
self.font_options = ["Ariel", "Times New Roman", "Helvetica

()self.blur_effect = QGraphicsBlurEffect
self.blur_effect.setBlurRadius(90) # Adjust the blur radius as needed
()screen = QDesktopWidget().screenGeometry
    Extract the screen width and height #
    ()self.screen_width = screen.width
    ()self.screen_height = screen.height

```

```
self.is_create_group_pressed = False
self.is_create_group_inside_chat_pressed = False
```

```
self.is_rename_group_pressed = False
self.is_add_users_to_group_pressed = False
```

```
self.current_search_song_dict = None
self.phone_number = None
self.email = None
self.messages_font_size = 12
```

```
self.volume = 50
"" = self.output_device_name
"" = self.input_device_name
self.camera_index = 0
self.font_size = 12
[self.font_text = self.font_options[0
"self.background_color = "Blue
self.censor_data_from_strangers = True
self.is_private_account = True
self.push_to_talk_key = None
self.two_factor_authentication = False
```

```
self.playlist_volume = self.volume
[] = self.playlist_songs
self.playlist_index = 0
self.playlist_last_index = 0
self.shuffle = False
```

```

        self.replay_song = False

        self.size_error_label = False
        self.is_chat_box_full = False
        self.is_friends_box_full = False
        self.is_last_message_on_screen = False

        [] = self.list_messages
        self.is_user_have_current_chat_all_messages = False

        [] = self.request_list
        self.is_in_a_call = False
        self.is_calling = False
        "" = self.calling_to
        "" = self.in_call_with
        self.is_getting_called = False
        "" = self.getting_called_by
        self.is_joining_call = False
        "" = self.joining_to
        [] = self.call_dicts

        "" = self.username

        "" = self.selected_chat
        self.is_current_chat_a_group = False

        ()self.camera_devices_names = get_camera_names

```



```

        self.mute = False
        self.deafen = False

        "self.selected_settings = "My Account
            self.is_push_to_talk = False
            self.is_push_to_talk_pressed = False
        self.is_editing_push_to_talk_button = False
            self.profile_pic = None
            [] = self.list_user_profile_dicts
        [] = self.circular_images_dicts_list_of_users
        [] = self.circular_images_dicts_list_of_groups

        self.is_watching_video = False
the scroll widget that contain all of the messages #
        self.messages_content_saver = None
        self.is_messages_need_update = True

        [] = self.online_users_list
            [] = self.friends_list
friend_box_page could be online, add friend, blocked, all, pending #
        "self.friends_box_page = "online
            [] = self.chats_list
            self.file_to_send = None
            "" = self.file_name
            self.chat_start_index = None
            self.Network = Network
            self.chat_box_chats_index = 0
        self.chat_box_index_y_start = 100

```

```

        self.friends_box_index = 0
        self.friends_box_index_y_start = 100
        self.current_friends_box_search = False

        [] = self.selected_group_members
        self.create_group_index = 0
        self.add_users_to_group_index = 0
        self.group_max_members = 10

        [] = self.blocked_list
        [] = self.groups_list

        self.chat_clicked_var = True
        self.social_clicked_var = False
        self.setting_clicked = False
        self.music_clicked = False

        self.is_new_chat_clicked = True

        self.current_chat_box_search = False
        [] = self.temp_search_list
        self.spacer = QSpacerItem(1, 1, QSizePolicy.Expanding,
                                   QSizePolicy.Expanding)

        (self.updated_chat_signal.connect(self.updated_chat
        (self.updated_social_page_signal.connect(self.updated_social_page
        (self.updated_settings_signal.connect(self.updated_settings_page
        (self.getting_call_signal.connect(self.getting_a_call
        (self.stop_sound_signal.connect(self.stop_sound

```

```

        (self.initiating_call_signal.connect(self.initiate_call
        (self.reset_call_var_signal.connect(self.reset_call_var

    (self.new_message_play_audio_signal.connect(self.new_message_play_audio

    (self.stop_watching_stream_signal.connect(self.stop_watching_video_stream

self.caching_circular_images_of_users_signal.connect(self.caching_circular_ima
    (ges_of_users

self.caching_circular_images_of_groups_signal.connect(self.caching_circular_im
    (ages_of_groups

(self.disconnect_signal.connect(self.page_controller_object.quit_application
    (self.updating_profile_dict_signal.connect(self.update_profile_dict_of_user
(self.update_group_lists_by_group.connect(self.update_groups_list_by_dict
    (self.update_message_box_signal.connect(self.update_message_box

self.scroll_back_to_index_before_update_signal.connect(self.scroll_back_to_ind
    (ex_before_update

self.insert_messages_into_message_box_signal.connect(self.insert_messages_i
    (nto_message_box

self.insert_new_message_in_chat_signal.connect(self.insert_new_message_in_
    (chat

    (self.insert_search_result_signal.connect(self.insert_search_result
        (self.close_call_threads_signal.connect(self.close_call_threads
            (self.start_call_threads_signal.connect(self.start_call_threads
                (self.insert_playlist_to_table_signal.connect(self.insert_playlist_to_table

    (self.update_settings_from_dict_signal.connect(self.update_settings_from_dict

```

```
self.update_chat_page_without_messages_signal.connect(self.update_chat_pag
(e_without_messages
```

```
    )self.sound_effect_media_player = QMediaPlayer
    (self.sound_effect_media_player.setVolume(50
```

```
    )self.mp3_message_media_player = QMediaPlayer
    (self.mp3_message_media_player.setVolume(50
```

```
    )self.calling_media_player = QMediaPlayer
    (self.calling_media_player.setVolume(50
```

```
    )self.playlist_media_player = QMediaPlayer
    (self.playlist_media_player.setVolume(50
```

```
self.playlist_media_player.mediaStatusChanged.connect(self.handle_playlist_so
(ng_state_change
```

```
(self.playlist_media_player.positionChanged.connect(self.update_slider_position
```

```
    )self.ringtone_media_player = QMediaPlayer
```

```
self.ringtone_media_player.stateChanged.connect(self.handle_state_changed_s
(ound_effect
```

```
(self.ringtone_media_player.setVolume(50
```

```

self.ringtone =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Getting_called_sound_
(('effect.mp3
```

```

        self.ding_sound_effect =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Ding Sound
                                (('Effect.mp3

        self.new_message_audio =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/new_message_sound_
                                (('effect.mp3

        (self.sound_effect_media_player.setMedia(self.ringtone
        ,(x, y = (int(self.page_controller_object.screen_width * 0.052
        ((int(self.page_controller_object.screen_height * 0.092
        ,(width, height = (int(self.page_controller_object.screen_width * 0.3125
        ((int(self.page_controller_object.screen_height * 0.37

        (self.setGeometry(x, y, width, height
        ('self.setWindowTitle('Main Page
        ""self.setStyleSheet(f
                                }} QWidget
        ;{background-color: {self.background_color_hex
                                {{
                                (""

        Create an instance of ChatBox #
        :if self.chat_clicked_var

self.chat_box = ChatBox(self.list_messages, parent=self,
                        (Network=self.Network

        ()buttons_layout = QHBoxLayout
        (self.main_layout = QVBoxLayout(self
        (self.main_layout.addSpacing(30
        (self.main_layout.addLayout(buttons_layout
        ,self.friends_box = FriendsBox(friends_list=self.friends_list

```

```

requests_list=self.request_list, Network=self.Network,
                                ,username=self.username
                                (parent=self
                                ()self.friends_box.hide
                                (self.settings_box = SettingsBox(parent=self
                                ()self.settings_box.hide
                                (self.music_box = PlaylistWidget(main_page_widget=self
                                ()self.music_box.hide
                                (self.stacked_widget = QStackedWidget(self
                                (self.stacked_widget.addWidget(self.chat_box
self.stacked_widget.addWidget(self.settings_box) # Placeholder for the
                                Settings page
                                (self.stacked_widget.addWidget(self.friends_box
                                (self.stacked_widget.addWidget(self.music_box
                                (self.main_layout.addWidget(self.stacked_widget

                                (self.setLayout(self.main_layout

                                :(def close_all_threads(self
                                turns every thread flag to False #
                                :try
                                self.vc_thread_flag = False
                                self.vc_play_flag = False
                                self.listen_udp = False
                                self.is_screen_shared = False
                                self.is_camera_shared = False
                                :except Exception as e
                                ("{print(f"error in closing threads {e

```

```

                                : (def thread_play_vc_data(self
                                : try
                                ("....print("started play voice data thread
output_device_name = self.output_device_name # Get the output device
                                                name from main_page

                                : if output_device_name == "Default
                                ()def_device_name = get_default_output_device_name
                                (output_device_index = find_output_device_index(def_device_name
                                                : else
                                                output_device_index =
                                                (find_output_device_index(output_device_name

                                : if output_device_index is None
                                (.print(f"Output device '{output_device_name}' not found
                                                : else
                                ("{print(f"Output index is {output_device_name

                                output_stream = p.open(format=FORMAT, channels=CHANNELS,
                                                ,rate=RATE, output=True, frames_per_buffer=CHUNK
                                (output_device_index=output_device_index
                                                : while self.vc_play_flag
                                                : try
                                                : if len(self.vc_data_list) > 0
                                                ()self.audio_data_lock.acquire
                                all_audio_data = [np.frombuffer(vc_data, dtype=np.int16) for
                                                [vc_data, _ in self.vc_data_list
                                                mixed_data =
                                                (np.vstack(all_audio_data).mean(axis=0).astype(np.int16
                                ()mixed_data_audio_bytes = mixed_data.tobytes

```

```

self.vc_data_list = [] # Clear the vc_data_list after processing

        modified_audio_data_list =
(audio_data_list_set_volume([mixed_data_audio_bytes], volume=self.volume
        (modified_data = b"".join(modified_audio_data_list
        Play the modified audio data #
        (output_stream.write(modified_data
        ()self.audio_data_lock.release
        :except Exception as e
        pass # Handle the case where the queue is empty
        (time.sleep(0.02
        ()output_stream.stop_stream
        ()output_stream.close
        :except Exception as e
        ("{print(f"error in thread_play_vc_data {e

        :(def thread_send_voice_chat_data(self
        n = self.Network
        :try
        [] = accumulated_data
        ("....print("started voice chat thread
input_device_name = self.input_device_name # Get the output device
        name from main_page

        :if input_device_name == "Default" or input_device_name == "default
        ()def_device_name = get_default_input_device_name
        (input_device_index = find_input_device_index(def_device_name
        :else
        (input_device_index = find_input_device_index(input_device_name

```



```

        :if input_device_index is None
        (.print(f'Input device '{input_device_name}' not found
        ("{"print(f"input index is {input_device_index

        ,input_stream = p.open(format=FORMAT
        ,channels=CHANNELS
        ,rate=RATE
        ,input=True
        ,frames_per_buffer=CHUNK
        (input_device_index=input_device_index

        :while self.vc_thread_flag
        :if not self.mute and not self.deafen
        :if self.is_push_to_talk
        :if self.is_push_to_talk_pressed
        (input_data = input_stream.read(CHUNK
        (n.send_vc_data(input_data
        :else
        (time.sleep(0.1
        :else
        (input_data = input_stream.read(CHUNK
        (n.send_vc_data(input_data
        :else
        (time.sleep(0.1
        ()input_stream.stop_stream
        ()input_stream.close
        ("....print("stopped voice chat thread

```

```

                                :except Exception as e
({print(f"error in thread_send_voice_chat_data {e

                                :(def thread_send_share_screen_data(self
                                time_between_frame = 1 / SCREEN_FPS
                                :try
                                :while self.is_screen_shared
                                Capture the screen using PyAutoGUI #
                                ()screen = ImageGrab.grab
                                Convert the screenshot to a NumPy array #
                                (frame = np.array(screen
                                (frame_bgr = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR

                                ()frame_bytes = frame_bgr.tobytes
                                Send the frame to the server #
                                (self.Network.send_share_screen_data(frame_bytes, frame.shape

                                time.sleep(time_between_frame) # Adjust the sleep time based on
                                                                your needs
                                ("print("send share screen data thread closed
                                :except Exception as e
                                ("{print(f"Screen sharing error: {e

                                :(def thread_send_share_camera_data(self
                                time_between_frame = 1 / CAMERA_FPS
                                network = self.Network
                                :try
                                Initialize the camera #

```

```
cap = cv2.VideoCapture(self.camera_index) # Use 0 for default camera,  
                                           change as needed
```

```
(set_camera_properties(cap
```

```
:while self.is_camera_shared
```

```
Capture frame-by-frame #
```

```
()ret, frame = cap.read
```

```
:if not ret
```

```
("print("Error: Couldn't capture frame from camera
```

```
break
```

```
Convert the frame to a NumPy array #
```

```
(frame_np = np.asarray(frame
```

```
Convert the NumPy array to bytes #
```

```
()frame_bytes = frame_np.tobytes
```

```
Send the frame to the server #
```

```
(network.send_share_camera_data(frame_bytes, frame_np.shape
```

```
time.sleep(time_between_frame) # Adjust the sleep time based on  
                                your needs
```

```
Release the camera and close thread #
```

```
()cap.release
```

```
("print("send share camera data thread closed
```

```
:except Exception as e
```

```
("{print(f"Camera sharing error: {e
```

```
:(def listen_udp_socket_thread(self
```

```
("print("started listen_udp thread
```

```

        network = self.Network
        :while self.listen_udp
            :try
                ()fragment_data, address = network.recv_udp
                :if fragment_data
                    (data = pickle.loads(fragment_data
                    (self.handle_udp_data(data
                    :except OSError as os_err
                    ("){print(f"OS error: {os_err
                    :except Exception as e
                    ("){print(f"Exception: {e

                :(def handle_udp_data(self, data
                ("message_type = data.get("message_type
                :try

                :("if message_type == "vc_data
                    (self.handle_vc_data(data
                :("elif message_type in ["share_screen_data", "share_camera_data
                    (self.handle_stream_data(data
                    :except Exception as e
                    ("){print(f"Error in rebuilding UDP packets: {e

                :(def handle_vc_data(self, data
                    ("is_last = data.get("is_last
                    ("is_first = data.get("is_first
                    data_list = self.vc_data_fragments_list

                ("self.handle_data_fragment(data_list, data, is_first, is_last, "vc_data

```

```

        : (def handle_stream_data(self, data
            ("is_last = data.get("is_last
            ("is_first = data.get("is_first
        )data_list = self.share_screen_data_fragments_list if data.get
        message_type") == "share_screen_data" else"
        self.share_camera_data_fragments_list

("self.handle_data_fragment(data_list, data, is_first, is_last, "stream_data

:(def handle_data_fragment(self, data_list, data, is_first, is_last, data_type
        :if is_first
        ()data_list.clear

        (("data_list.append(data.get("sliced_data

        :if is_last
        (compressed_data = b".join(data_list
        ("speaker = data.get("speaker
        shape_of_frame = data.get("shape_of_frame") if data_type ==
        "stream_data" else None

        (decompressed_data = zlib.decompress(compressed_data
        : "if data_type == "stream_data
        decompressed_frame = np.frombuffer(decompressed_data,
        (dtype=np.uint8).reshape(shape_of_frame
        (self.update_stream_screen_frame(decompressed_frame
        : "elif data_type == "vc_data
        ()self.audio_data_lock.acquire

```

```

((self.vc_data_list.append((decompressed_data, speaker
                                ()self.audio_data_lock.release

                                ()data_list.clear

                                : (def exit_group(self, group_id
                                : try
                                :
                                (group_name = self.get_group_name_by_id(group_id
                                "{group_name_plus_id = f"({group_id}){group_name
                                (self.chats_list.remove(group_name_plus_id
                                : if self.selected_chat == group_name_plus_id
                                "" = self.selected_chat
                                [] = self.list_messages
                                (self.Network.send_exit_group(group_id
                                ()self.updated_chat
                                : except Exception as e
                                ("{print(f"error in existing group {e

                                : (def remove_friend(self, chat
                                (self.friends_list.remove(chat
                                ()self.updated_social_page
                                ()self.update_chat_page_without_messages
                                (self.Network.send_remove_chat(chat

                                : (def send_friend_request_for_user(self, user
                                (self.Network.send_friend_request(user

                                : (def remove_user_from_group(self, user, group_id

```

```

(self.Network.send_remove_user_from_group(user, group_id

def right_click_object_func(self, pos, parent, button, actions_list,
                           :(chat_name=None, group_id=None
                               :try
                                   (menu = QMenu(parent
                                       ""menu.setStyleSheet(f
                                           }} QMenu
/* color: white; /* Text color of menu items
/* border: 1px solid gray; /* Border style
                                   {{
                                       }} QMenu::item:selected
background-color: {self.standard_hover_color}; /* Hover color when
                                           /* item is selected
                                   {{
                                       (""
                                           :for item1 in actions_list
(((" ", "_")action = menu.addAction(item1.replace
                                   :if item1 == "remove_chat
((action.triggered.connect(lambda: self.remove_friend(chat_name
                                   :elif item1 == "exit_group
((action.triggered.connect(lambda: self.exit_group(group_id
                                   :elif item1 == "add_friend
                                   action.triggered.connect(lambda:
                                   ((self.send_friend_request_for_user(chat_name
                                   :elif item1 == "remove_user_from_group
                                   action.triggered.connect(lambda:
                                   ((self.remove_user_from_group(chat_name, group_id
                                   :elif item1 == "message_user

```

```

        action.triggered.connect(lambda:
            ((self.chat_box.selected_chat_changed(chat_name
Use the position of the button as the reference for menu placement #
            (global_pos = button.mapToGlobal(pos

```

```

        Show the context menu at the adjusted position #
            (menu.exec_(global_pos
                :except Exception as e
            ("{{print(f"error in right click func {e

```

```

            :(def update_slider_position(self, new_position
        (self.music_box.playlist_duration_slider.setValue(new_position
            (current_time = QTime(0, 0).addMSecs(new_position
                ("current_time_str = current_time.toString("mm:ss
        (self.music_box.update_current_duration_text(current_time_str

```

```

            :(def toggle_shuffle(self
                :if self.shuffle
                self.shuffle = False
                :else
                self.shuffle = True

```

```

            :(def remove_song_from_playlist(self
                :try
                index = self.playlist_index
                [song_dict = self.playlist_songs[index
                [del self.playlist_songs[index
                ("song_title = song_dict.get("title
        (self.Network.send_remove_song_from_playlist(song_title

```



```

:if not self.playlist_media_player.state() == QMediaPlayer.PausedState
    ()self.play_playlist_by_index
    :else
    ()self.play_playlist_by_index
    ()self.playlist_media_player.pause
    ("{}print(f"deleted song of index {index
    :except Exception as e
    ("{}print(f"Couldn't remove song {e

:(def music_button_clicked(self
    self.chat_clicked_var = False
    self.social_clicked_var = False
    self.setting_clicked = False
    self.music_clicked = True
    (self.stacked_widget.setCurrentIndex(3

:(def insert_search_result(self, result_dict
    ()self.play_ding_sound_effect
    self.current_search_song_dict = result_dict
    (self.music_box.insert_search_data(result_dict

:(def play_search_result(self
:if self.current_search_song_dict is not None
    current_search_audio_bytes =
    ("self.current_search_song_dict.get("audio_bytes
    (self.music_box.playlist_duration_slider.setMinimum(0
    ("duration_str = self.current_search_song_dict.get("audio_duration
    (self.music_box.update_duration_tex(duration_str
    (total_duration = duration_to_milliseconds(duration_str

```

```

(self.music_box.playlist_duration_slider.setMaximum(total_duration
    play_mp3_from_bytes(current_search_audio_bytes,
                        (self.playlist_media_player

:(def save_searched_song_to_playlist(self
    :try
    :if self.current_search_song_dict is not None
    :(if not self.is_song_exist_in_playlist(self.current_search_song_dict
    (self.Network.save_song_in_playlist(self.current_search_song_dict
        (temp_dict = self.current_search_song_dict.copy
            :if "audio_bytes" in temp_dict
            ["del temp_dict["audio_bytes
            (self.playlist_songs.append(temp_dict

(self.music_box.insert_new_song_to_playlist(self.current_search_song_dict
    ("print("send song to server
    :else
    ("print("song already exist in playlist
    :except Exception as e
    ("{print(f"error in adding song to playlist {e

:(def is_song_exist_in_playlist(self, added_dict
    :try
    ("added_dict_title = added_dict.get("title
    :for song in self.playlist_songs
    ("song_title = song.get("title
    :if song_title == added_dict_title
        return True
        return False

```

```

                                :except Exception as e
                                ("{print(f"error in finding duplicates {e

                                :(def insert_playlist_to_table(self
                                (self.music_box.insert_playlist_songs(self.playlist_songs

                                :(def pause_and_unpause_playlist(self
:if self.playlist_media_player.state() == QMediaPlayer.PlayingState
                                ()self.playlist_media_player.pause
:elif self.playlist_media_player.state() == QMediaPlayer.PausedState
                                ()self.playlist_media_player.play
:elif self.playlist_media_player.state() == QMediaPlayer.StoppedState
                                Handle the case where there is no media loaded #
                                ("print("No media loaded in the playlist media player
                                :if len(self.playlist_songs) > 0
                                ()self.play_playlist_by_index

                                :(def go_to_last_song(self
                                :if not self.shuffle
                                :if self.playlist_index - 1 < 0
                                pass
                                :else
                                (self.set_new_playlist_index_and_listen(self.playlist_index - 1
                                :else
                                (self.set_new_playlist_index_and_listen(self.playlist_last_index

                                :(def go_to_next_song(self
                                :try

```

```

        (len_songs_list = len(self.playlist_songs
                                :if not self.shuffle
                                :if self.playlist_index + 1 < len_songs_list
(self.set_new_playlist_index_and_listen(self.playlist_index + 1
                                :else
                                (self.set_new_playlist_index_and_listen(0
                                :else
                                (random_index = random.randint(0, len(self.playlist_songs) - 1
                                :while random_index == self.playlist_index
(random_index = random.randint(0, len(self.playlist_songs) - 1
                                (self.set_new_playlist_index_and_listen(random_index
                                :except Exception as e
                                ("{print(f"error in going to next song {e

:(def handle_playlist_song_state_change(self, status
                                :if status == QMediaPlayer.EndOfMedia
                                :if not self.replay_song
                                :if not self.shuffle
                                (len_songs_list = len(self.playlist_songs
                                :if self.playlist_index + 1 < len_songs_list
(self.set_new_playlist_index_and_listen(self.playlist_index + 1
                                :else
                                (self.set_new_playlist_index_and_listen(0
                                :else
                                (random_index = random.randint(0, len(self.playlist_songs) - 1
                                :while random_index == self.playlist_index
(random_index = random.randint(0, len(self.playlist_songs) - 1
                                (self.set_new_playlist_index_and_listen(random_index

```

```

:else
(self.set_new_playlist_index_and_listen(self.playlist_index

:(def set_new_playlist_index_and_listen(self, index
    self.playlist_last_index = self.playlist_index
        self.playlist_index = index
    ()self.play_playlist_by_index

:(def play_playlist_by_index(self
    (len_songs_list = len(self.playlist_songs
        :if self.playlist_index < len_songs_list
    ("{print(f'trying to listen to song of index {self.playlist_index
        (self.music_box.select_row(self.playlist_index
(self.Network.ask_for_song_bytes_by_playlist_index(self.playlist_index
:else
    self.playlist_index = 0
    ("{print(f'trying to listen to song of index {self.playlist_index
        (self.music_box.select_row(self.playlist_index
(self.Network.ask_for_song_bytes_by_playlist_index(self.playlist_index
    (self.music_box.playlist_duration_slider.setMinimum(0
    [song_dict = self.playlist_songs[self.playlist_index
        ("duration_str = song_dict.get("audio_duration
    (self.music_box.update_duration_tex(duration_str
    (total_duration = duration_to_milliseconds(duration_str
(self.music_box.playlist_duration_slider.setMaximum(total_duration

:(def get_setting_dict(self
    } = settings_dict

```

```

        ,volume": self.volume"
        ,output_device": self.output_device_name"
        ,input_device": self.input_device_name"
        ,camera_device_index": self.camera_index"
        ,font_size": self.font_size"
        ,font": self.font_text"
        ,theme_color": self.background_color"
        ,censor_data": self.censor_data_from_strangers"
        ,private_account": self.is_private_account"
        ,push_to_talk_bind": self.push_to_talk_key"
        two_factor_auth": self.two_factor_authentication"
    }
    return settings_dict

```

```

:(def update_settings_from_dict(self, settings_dict
    Update settings from the provided dictionary #
    ("self.volume = settings_dict.get("volume
        self.playlist_volume = self.volume
    ("self.output_device_name = settings_dict.get("output_device
        ("self.input_device_name = settings_dict.get("input_device
    ("self.camera_index = settings_dict.get("camera_device_index
        ("self.font_size = settings_dict.get("font_size
            ("self.font_text = settings_dict.get("font
    ("self.update_background_color(settings_dict.get("theme_color
    ("self.censor_data_from_strangers = settings_dict.get("censor_data
        ("self.is_private_account = settings_dict.get("private_account
        ("self.push_to_talk_key = settings_dict.get("push_to_talk_bind
    ("self.two_factor_authentication = settings_dict.get("two_factor_auth

```

```

        ()self.updated_settings_page
        ()self.updated_chat

        : (def update_settings_dict(self
        ()settings_dict = self.get_setting_dict
        (self.Network.send_settings_dict_to_server(settings_dict

        : (def start_listen_udp_thread(self
        ()self.listen_udp_thread.start

        : (def start_call_threads(self
        [] = self.vc_data_list
        ()self.start_send_vc_thread
        ()self.start_listen_thread

        : (def close_call_threads(self
        [] = self.vc_data_list
        ()self.close_listen_thread
        ()self.close_send_vc_thread

        : (def close_listen_thread(self
        self.vc_play_flag = False
        ()self.play_vc_data_thread.join
        self.play_vc_data_thread =
        ((=threading.Thread(target=self.thread_play_vc_data, args

        : (def close_send_vc_thread(self
        self.vc_thread_flag = False
        ()self.send_vc_data_thread.join

```

```

        self.send_vc_data_thread =
        (())=threading.Thread(target=self.thread_send_voice_chat_data, args

        : (def start_listen_thread(self
            self.vc_play_flag = True
        ()self.play_vc_data_thread.start

        : (def start_send_vc_thread(self
            self.vc_thread_flag = True
        ()self.send_vc_data_thread.start

        : (def scroll_back_to_index_before_update(self, n_last_widgets
        (self.messages_content_saver.scroll_up_by_n_widgets(n_last_widgets
            ("print(f"scrolled back down by {n_last_widgets} widgets

        : (def insert_messages_into_message_box(self, messages_list

        (self.messages_content_saver.insert_messages_list_to_layout(messages_list

        : (def insert_new_message_in_chat(self, message_dict
        (self.messages_content_saver.add_new_message_at_start(message_dict

        : (def insert_message_that_client_send(self, message_dict
            (self.insert_new_message_in_chat(message_dict
                ()self.scroll_maximum_down

        : (def scroll_maximum_down(self
            ()self.messages_content_saver.scroll_maximum

```



```

                                :(def update_message_box(self
                                ()self.messages_content_saver.update_messages_layout

                                :(def update_groups_list_by_dict(self, updated_group_dict
                                index = 0
                                :for group_dict in self.groups_list
                                :("if group_dict.get("group_id") == updated_group_dict.get("group_id
                                self.groups_list[index] = updated_group_dict
                                ("{'print(f"updated group dict of id {updated_group_dict.get('group_id
                                new_image =
                                (('base64.b64decode(updated_group_dict.get('group_b64_encoded_image

                                self.update_circular_photo_of_group(updated_group_dict.get('group_id'),
                                (new_image

                                ()self.update_chat_page_without_messages

                                return

                                index += 1

                                :(def update_media_players_volume(self, value
                                (self.mp3_message_media_player.setVolume(value
                                (self.sound_effect_media_player.setVolume(value
                                (self.playlist_media_player.setVolume(value #
                                (self.ringtone_media_player.setVolume(value
                                (self.calling_media_player.setVolume(value

                                :(def pause_or_unpause_mp3_files_player(self
                                :if self.mp3_message_media_player.state() == QMediaPlayer.PlayingState
                                ()self.mp3_message_media_player.pause
                                elif self.mp3_message_media_player.state() ==
                                :QMediaPlayer.PausedState

```

```

        ()self.mp3_message_media_player.play

        : (def update_every_screen(self
            : try
                ()self.updated_chat
                ()self.updated_social_page
                ()self.updated_settings_page
            : except Exception as e
                ("{print(f"error in updating screens: {e

: (def update_background_color(self, new_background_color_str
            background_color_hex =
            (self.color_design_mapping.get(new_background_color_str
            self.background_color = new_background_color_str
            self.background_color_hex = background_color_hex
            get_new_hover_color =
            (self.style_color_hover_mapping.get(new_background_color_str
            self.standard_hover_color = get_new_hover_color
            ""self.setStyleSheet(f
                }} QWidget
            ;{background-color: {background_color_hex
                {{
                (""
            ()self.music_box.update_music_page_style_sheet
            ("print("updated background color
            : try
                ()self.update_every_screen
            : except Exception as e
                ("{print(f"error in changing background color: {e

```

```

        : (def get_circular_image_bytes_by_name(self, name
:for circular_image_dictionary in self.circular_images_dicts_list_of_users
        ("username = circular_image_dictionary.get("username
            :if username == name
        ("return circular_image_dictionary.get("circular_image_bytes
            return None

        : (def get_circular_image_bytes_by_group_id(self, group_id
:for circular_image_dictionary in self.circular_images_dicts_list_of_groups
        ("current_group_id = circular_image_dictionary.get("group_id
            :if current_group_id == group_id
        ("return circular_image_dictionary.get("circular_image_bytes

        : (def caching_circular_images_of_users(self
            [] = self.circular_images_dicts_list_of_users
        :for profile_dictionary in self.list_user_profile_dicts
            ("username = profile_dictionary.get("username
        ("image_bytes_encoded = profile_dictionary.get("encoded_image_bytes
            :if image_bytes_encoded is None
                circular_image_bytes = None
            :else
                (image_bytes = base64.b64decode(image_bytes_encoded
        (circular_image_bytes = make_circular_image(image_bytes
            } = circular_images_dict
            ,username": username"
        circular_image_bytes": circular_image_bytes"

```

```

{
(self.circular_images_dicts_list_of_users.append(circular_images_dict
()self.update_chat_page_without_messages
()self.updated_settings_page

:(def caching_circular_images_of_groups(self
[] = self.circular_images_dicts_list_of_groups
:for group in self.groups_list
("group_id = group.get("group_id
("image_bytes_encoded = group.get("group_b64_encoded_image
:if image_bytes_encoded is None
circular_image_bytes = None
:else
(image_bytes = base64.b64decode(image_bytes_encoded
(circular_image_bytes = make_circular_image(image_bytes
} = circular_images_dict
,group_id": group_id"
circular_image_bytes": circular_image_bytes"

{
(self.circular_images_dicts_list_of_groups.append(circular_images_dict
()self.update_chat_page_without_messages

:(def update_profile_dict_of_user(self, name, new_profile_dict
:try
index = 0
:for profile_dict in self.list_user_profile_dicts
:if profile_dict.get("username") == name

```

```

        self.list_user_profile_dicts[index] = new_profile_dict
    ("encoded_image = new_profile_dict.get("encoded_image_bytes
        :if encoded_image is not None
        self.update_circular_photo_of_user(name,
            ((base64.b64decode(encoded_image
                :else
    (self.update_circular_photo_of_user(name, encoded_image
        break
        index += 1
    :except Exception as e
    ("{print(f"error in updating profile dict of user: {e

def update_profile_pic_dicts_list(self, name, new_image_bytes,
    :(circular_pic_bytes=None
    :for profile_dict in self.list_user_profile_dicts
    :if profile_dict.get("username") == name
    :if new_image_bytes is not None
    profile_dict["encoded_image_bytes"] =
        (base64.b64encode(new_image_bytes).decode
    :else
    profile_dict["encoded_image_bytes"] = None
    ("{print(f"updated the profile pic in dictionary list of username {name
    self.update_circular_photo_of_user(name, new_image_bytes,
        (circular_pic_bytes
        break

def update_circular_photo_of_user(self, username, new_photo,
    :(circular_pic_bytes=None
    :if new_photo is None
    circular_image = None

```

```

:else

        :if circular_pic_bytes is None
(circular_image = make_circular_image(new_photo
:else

        circular_image = circular_pic_bytes
        Iterate through the list of circular image dictionaries #
:for user_dict in self.circular_images_dicts_list_of_users
        Check if the username matches #
        :if user_dict["username"] == username
        Update the circular photo for the user #
        user_dict["circular_image_bytes"] = circular_image
        Exit the loop since the update is done #
        ({print(f"update_circular_photo_of_user of {username
        break

After updating, call the method to notify any listeners about the update #
        ()self.update_chat_page_without_messages

def update_circular_photo_of_group(self, group_id, new_photo,
        :(circular_pic_bytes=None
        :if new_photo is None
        circular_image = None
        :else
        :if circular_pic_bytes is None
(circular_image = make_circular_image(new_photo
        :else
        circular_image = circular_pic_bytes
        Iterate through the list of circular image dictionaries #
:for group_dict in self.circular_images_dicts_list_of_groups
        Check if the username matches #

```

```

        :if group_dict["group_id"] == group_id
        Update the circular photo for the user #
        group_dict["circular_image_bytes"] = circular_image
        Exit the loop since the update is done #
        ("{print(f"update_circular_photo_of_user of group id :{group_id
                                break
After updating, call the method to notify any listeners about the update #
        ()self.update_chat_page_without_messages

        :(def get_profile_pic_by_username(self, username
            :if self.list_user_profile_dicts is not None
            :for profile_dict in self.list_user_profile_dicts
            :if profile_dict.get("username") == username
("image_bytes_encoded = profile_dict.get("encoded_image_bytes
            :if image_bytes_encoded is not None
            (return base64.b64decode(image_bytes_encoded
                                :else
                                return None

        :(def set_page_index_by_clicked(self
            :if self.chat_clicked_var
            (self.stacked_widget.setCurrentIndex(0
            :elif self.social_clicked_var
            (self.stacked_widget.setCurrentIndex(2
            :elif self.setting_clicked
            (self.stacked_widget.setCurrentIndex(1

        :(def stop_watching_video(self

```

```

:try
    self.is_watching_video = False
    widget_to_remove = self.stacked_widget.currentWidget() # Get the
                                                              currently displayed widget
    (self.stacked_widget.removeWidget(widget_to_remove
    ()self.set_page_index_by_clicked
    ("print("exited video
    ()self.setFocus
    :except Exception as e
    ("{print(f"error in stopping video: {e

```

```

:(def start_watching_video(self, video_bytes
    :try
        self.is_watching_video = True
        (video_player = VideoPlayer(video_bytes, self
        ()number_of_widgets = self.stacked_widget.count
        (self.stacked_widget.addWidget(video_player
        (self.stacked_widget.setCurrentIndex(number_of_widgets
        ()video_player.play_video
        :except Exception as e
        ("{print(f"had error trying to show video: {e

```

```

:(def start_share_screen_send_thread(self
    ()self.send_share_screen_thread.start
    ("print("Started Share screen thread

```

```

:(def start_camera_data_thread(self
    self.is_camera_shared = True
    ()self.send_camera_data_thread.start

```



```

("print("Started Share camera thread

:(def update_share_screen_thread(self
    self.send_share_screen_thread =
    (())=threading.Thread(target=self.thread_send_share_screen_data, args

:(def update_share_camera_thread(self
    self.send_camera_data_thread =
    (())=threading.Thread(target=self.thread_send_share_camera_data, args

:(def end_share_camera_thread(self
    self.is_camera_shared = False
    ()self.send_camera_data_thread.join
    self.send_camera_data_thread =
    (())=threading.Thread(target=self.thread_send_share_camera_data, args

:(def update_stream_screen_frame(self, frame
    :try
    (self.stream_screen.display_frame(frame
    :except Exception as e
    ("{print(f"update_stream_screen_frame: {e

:(def stop_watching_video_stream(self
    self.is_watching_screen = False
    "" = self.watching_user
    self.watching_type = None
    ()self.stream_screen.close
    ()self.showMaximized

```

```

        : (def start_watching_video_stream(self
        (self.stream_screen = VideoClient(self
        ()self.hide
        ()self.stream_screen.showMaximized

: (def get_group_manager_by_group_id(self, id
    : for group_dict in self.groups_list
    : if group_dict["group_id"] == id
("return group_dict.get("group_manager
    : else
        return None

: (def get_group_name_by_id(self, id
    : for group_dict in self.groups_list
    : if group_dict["group_id"] == id
("return group_dict.get("group_name
    : else
        return None

: (def is_call_dict_exist_by_group_id(self, group_id
    : for call_dict in self.call_dicts
    : ("if call_dict.get("is_group_call
: if call_dict.get("group_id") == group_id
        return True
        return False

: (def get_number_of_members_by_group_id(self, group_id
((return len(self.get_group_members_by_group_id(group_id

```

```

:(def get_group_members_by_group_id(self, group_id
                                   (group_id = int(group_id
                                   :for group in self.groups_list
                                   :if group["group_id"] == group_id
                                   ["return group["group_members
return None # Return 0 if the group ID is not found

:(def remove_call_dict_by_id(self, id_to_remove
                              :for call_dict in self.call_dicts
                              :if call_dict.get("call_id") == id_to_remove
                              (self.call_dicts.remove(call_dict
(print("removed call dict, because call isn't available anymore

:(def get_call_dict_by_group_id(self, group_id
                              :for call_dict in self.call_dicts
                              :("if call_dict.get("is_group_call
                              :if call_dict.get("group_id") == group_id
                              return call_dict
                              ("print("could not find the call dict

:(def get_call_dict_by_user(self, user
                              :for call_dict in self.call_dicts
                              :("if user in call_dict.get("participants
                              return call_dict
                              ("print("could not find the call dict

:(def is_call_dict_exists_by_id(self, call_id

```

```

        :for call_dict in self.call_dicts
        :if call_dict.get("call_id") == call_id
            return True
        return False

    :(def find_difference(self, list1, list2
        ((return list(set(list1) - set(list2

    :(def update_call_dict_by_id(self, updated_call_dict
        ("updated_participants = updated_call_dict.get("participants
            :for call_dict in self.call_dicts
                ("participants_before = call_dict.get("participants
            :("if call_dict.get("call_id") == updated_call_dict.get("call_id
                (self.call_dicts.remove(call_dict
            if len(updated_participants) > len(participants_before) and
                :self.username in updated_participants
        different_users = self.find_difference(updated_participants,
                                                (participants_before
    :if len(different_users) == 1 and self.username not in different_users
        join_sound =
    QMediaContent(QUrl.fromLocalFile('discord_app_assets/join_call_sound_effect.
                                                ('mp3
        (self.play_sound_effect(join_sound
    elif len(updated_participants) < len(participants_before) and
        :self.username in updated_participants
        )user_left_sound = QMediaContent
    (('QUrl.fromLocalFile('discord_app_assets/leave_call_sound_effect.mp3
        (self.play_sound_effect(user_left_sound
        (self.call_dicts.append(updated_call_dict

```

```

:(def reset_call_var(self
                                :try
self.is_joining_call = False
    "" = self.joining_to
self.is_in_a_call = False
    self.is_calling = False
    "" = self.calling_to
    "" = self.in_call_with
self.is_getting_called = False
    "" = self.getting_called_by
    self.mute = False
    self.deafen = False
    ()self.stop_sound
    ()self.updated_chat
self.is_screen_shared = False
self.is_camera_shared = False
    "" = self.watching_user
:if self.is_watching_screen
()self.stop_watching_video_stream

self.is_watching_screen = False
    :except Exception as e
    ("{print(f"reset_call_var error: {e

:(def end_current_call(self
                                :try
                                :try
                                ()self.Network.leave_call

```

```

        (...print(f'client hang up call123
                :except Exception as e
        ({print(f'end_current_call error: {e

:(def parse_group_caller_format(self, input_format
Define a regular expression pattern to capture the information #
('(\(([()^]))\([()^])\([()^]+pattern = re.compile(r'\([()^]

Use the pattern to match the input_format #
(match = pattern.match(input_format

                :if match

                Extract the matched groups #
                ((group_id = int(match.group(1
                ()group_name = match.group(2).strip
                ()group_caller = match.group(3).strip

return group_id, group_name, group_caller
                :else

                Return None if no match is found #
                return None

:(def initiate_call(self
        self.is_in_a_call = True
        self.is_calling = False
        self.is_getting_called = False
        self.is_joining_call = False
        ()self.stop_sound

```

```

:try
    :"" =! if self.calling_to
    :if "(" in self.calling_to
        self.in_call_with = self.calling_to
        ("{print(f'in call with {self.in_call_with
            "" = self.calling_to
        :else
            self.in_call_with = self.calling_to
            ("{print(f'in call with {self.in_call_with
                "" = self.calling_to
            :"" =! elif self.getting_called_by
            :if "(" in self.getting_called_by
                group_id, group_name, group_caller =
                (self.parse_group_caller_format(self.getting_called_by
self.in_call_with = "(" + str(group_id) + ")" + group_name
                ("{print(f'in call with {self.in_call_with
                    "" = self.getting_called_by
                :else
                    self.in_call_with = self.getting_called_by
                    ("{print(f'in call with {self.in_call_with
                        "" = self.getting_called_by
                    :"" =! elif self.joining_to
                        self.in_call_with = self.joining_to
                        ("{print(f'in call with {self.in_call_with
                            "" = self.joining_to
                        :except Exception as e
                            ("{print(f'error in initiating call is {e

:(def handle_state_changed_sound_effect(self, state

```

```

        :if state == QMediaPlayer.StoppedState
            :if self.is_getting_called
                (self.ringtone_media_player.setMedia(self.ringtone
                    ()self.ringtone_media_player.play

        : (def play_ding_sound_effect(self
            :try
                (self.play_sound_effect(self.ding_sound_effect
                    :except Exception as e
                        ("{print(f'::e

        : (def getting_a_call(self
            :try
                (self.ringtone_media_player.setMedia(self.ringtone
                    ()self.ringtone_media_player.play
                    :except Exception as e
                        ("{print(f'::e

        : (def new_message_play_audio(self
            :try
                (self.sound_effect_media_player.setMedia(self.new_message_audio
                    ()self.sound_effect_media_player.play
                    :except Exception as e
                        ("{print(f'::e

        : (def play_sound_effect(self, sound
            :try
                (self.sound_effect_media_player.setMedia(sound

```



```

        ()self.sound_effect_media_player.play
        :except Exception as e
            ("{print(f'::e

:(def play_calling_sound_effect(self, sound
        :try
(self.calling_media_player.setMedia(sound
        ()self.calling_media_player.play
        :except Exception as e
            ("{print(f'::e

:(def stop_sound(self
        :try
        ()self.sound_effect_media_player.stop
        ()self.ringtone_media_player.stop
        ()self.calling_media_player.stop
        :except Exception as e
            ("{print(f"Error stopping sound: {e

:(def chat_clicked(self
        :if not self.chat_clicked_var
self.current_friends_box_search = False
        self.current_chat_box_search = False
            [] = self.temp_search_list
            self.chat_clicked_var = True
(self.stacked_widget.setCurrentIndex(0
        :if self.setting_clicked
            ()self.update_settings_dict

```

```

        self.setting_clicked = False
        self.social_clicked_var = False

        : (def settings_clicked(self
                                : try
                                : if not self.setting_clicked
self.current_friends_box_search = False
        self.current_chat_box_search = False
        (self.stacked_widget.setCurrentIndex(1
            self.chat_clicked_var = False
            self.setting_clicked = True
            self.social_clicked_var = False
                                : except Exception as e
                                (print(e

        : (def social_clicked(self
                                : if not self.social_clicked_var
self.current_friends_box_search = False
        self.current_chat_box_search = False
            [] = self.temp_search_list
        (self.stacked_widget.setCurrentIndex(2
            self.chat_clicked_var = False
            self.setting_clicked = False
            self.social_clicked_var = True

        : (def updated_social_page(self
                                : try

        : "if self.friends_box_page == "add friend

```

```

        (self.stacked_widget.removeWidget(self.friends_box
,self.friends_box = FriendsBox(friends_list=self.friends_list
                                requests_list=self.request_list,
                                ,Network=self.Network
                                ,username=self.username
                                (parent=self
(self.stacked_widget.insertWidget(2, self.friends_box
                                :if self.social_clicked_var
                                (self.stacked_widget.setCurrentIndex(2
                                :else
                                ()search_bar_text = self.friends_box.search.text
()has_had_focus_of_search_bar = self.friends_box.search.hasFocus
                                (self.stacked_widget.removeWidget(self.friends_box
,self.friends_box = FriendsBox(friends_list=self.friends_list
                                requests_list=self.request_list,
                                ,Network=self.Network
                                ,username=self.username
                                (parent=self
(self.stacked_widget.insertWidget(2, self.friends_box

                                :if len(search_bar_text) > 0
(self.friends_box.search.setText(search_bar_text
                                (self.friends_box.search.setFocus(True
                                ()self.friends_box.search.deselect
                                :else
                                :if has_had_focus_of_search_bar
(self.friends_box.search.setFocus(True
                                ()self.friends_box.search.deselect

```

```

        :if self.social_clicked_var
        (self.stacked_widget.setCurrentIndex(2
        :except Exception as e
        ("{print(f"error in updating social page{e

        :(def updated_settings_page(self
        :try
        (self.stacked_widget.removeWidget(self.settings_box
        (self.settings_box = SettingsBox(parent=self
        (self.stacked_widget.insertWidget(1, self.settings_box

        ()self.set_page_index_by_clicked
        :except Exception as e
        ("{print(f"error in updated_settings_page error:{e

        :(def keyReleaseEvent(self, event
        :if self.push_to_talk_key is not None
        ()key = event.key
        key_string = chr(key) if 32 <= key <= 126 else
        (self.special_keys_mapping.get(key
        :if key_string == self.push_to_talk_key
        :if self.is_push_to_talk
        self.is_push_to_talk_pressed = False
        ("print("push to talk released

        :(def keyPressEvent(self, event
        n = self.Network
        :if self.push_to_talk_key is not None
        ()key = event.key

```

```

        key_string = chr(key) if 32 <= key <= 126 else
            (self.special_keys_mapping.get(key
                :if key_string == self.push_to_talk_key
                    :if self.is_push_to_talk
                        self.is_push_to_talk_pressed = True
                        ("print("push to talk button toggled
                    :if event.key() == Qt.Key_Return or event.key() == Qt.Key_Enter
                        :try
if self.chat_clicked_var and self.chat_box.chat_name_label.text() != ""
                        :or self.setting_clicked
                            :()if self.chat_box.check_editing_status
                                :if len(self.chat_box.text_entry.text()) > 0
                                    ()current_time = datetime.datetime.now
('formatted_time = current_time.strftime('%Y-%m-%d %H:%M
                                message_dict =
                                ,create_message_dict(self.chat_box.text_entry.text(), self.username
('str(formatted_time), "string", None
                                (self.list_messages.insert(0, message_dict
n.send_message(self.username, self.selected_chat,
                                , "message_dict.get("content")", "string
                                (None
                                ("print("Sent message to server
                                ("")self.chat_box.text_entry.setText
                                :[if self.selected_chat != self.chats_list[0
                                    (self.chats_list.remove(self.selected_chat
                                    (self.chats_list.insert(0, self.selected_chat
                                    ()self.chat_box.text_entry.setFocus
(self.insert_message_that_client_send(message_dict
                                self.is_new_chat_clicked = True
                                ()self.updated_chat

```

```

(self.chat_box.text_entry.setFocus(True
                                :if self.file_to_send
,Compresses the byte representation of an image using zlib #
encodes the compressed data as base64, and then decodes #
.it into a UTF-8 string for transmission or storage #
(compressed_byte_file = zlib.compress(self.file_to_send
                                compressed_base64_file =
                                ()base64.b64encode(compressed_byte_file).decode
                                ((print(len(compressed_base64_image #
                                ()current_time = datetime.datetime.now
('formatted_time = current_time.strftime('%Y-%m-%d %H:%M
                                "" = file_type
                                :(("if self.file_name.endswith(("png", "jpg
                                "file_type = "image
                                :(("elif self.file_name.endswith(("mp4", "mov
                                "file_type = "video
                                :("elif self.file_name.endswith("mp3
                                "file_type = "audio
                                :("elif self.file_name.endswith("txt
                                "file_type = "txt
                                :("elif self.file_name.endswith("pdf
                                "file_type = "pdf
                                :("elif self.file_name.endswith("pptx
                                "file_type = "pptx
                                :("elif self.file_name.endswith("docx
                                "file_type = "docx
                                :("elif self.file_name.endswith("py
                                "file_type = "py
                                :("elif self.file_name.endswith("xlsx

```

```

        "file_type = "xlsx
message_dict = create_message_dict(compressed_base64_file,
                                   ,self.username
(str(formatted_time), file_type, self.file_name
        (self.list_messages.insert(0, message_dict
add here that the type of the message is sent as well #
                                   :try
n.send_message(self.username, self.selected_chat,
               ,compressed_base64_file, file_type
               (self.file_name
               :except Exception as e
               ("print(f"error in sending message
               self.file_to_send = None
               "" = self.file_name
               :[if self.selected_chat != self.chats_list[0
               (self.chats_list.remove(self.selected_chat
               (self.chats_list.insert(0, self.selected_chat
               (self.insert_message_that_client_send(message_dict
               self.is_new_chat_clicked = True
               ()self.updated_chat
               (self.chat_box.text_entry.setFocus(True
               :elif self.social_clicked_var
               ()self.friends_box.send_friend_request
               :elif self.music_clicked
               ()search_str = self.music_box.search_song_entry.text
               :if len(search_str) > 0
               (n.send_song_search(search_str
               ("")self.music_box.search_song_entry.setText
               :except Exception as e

```

```

        ("{print(f"expection in key press event:{e
            :elif event.key() == Qt.Key_Escape
                :if not self.chat_clicked_var
                    ()self.chat_clicked
                    ()self.updated_chat
                :else
                    :if self.is_create_group_pressed
self.is_create_group_pressed = False
                    ()self.selected_group_members.clear
                    ()self.updated_chat
                :else
:if self.setting_clicked and self.is_editing_push_to_talk_button
                    ()key = event.key
key_string = chr(key) if 32 <= key <= 126 else
                    (self.special_keys_mapping.get(key
self.push_to_talk_key = key_string
self.is_editing_push_to_talk_button = False
                    ()self.updated_settings_page

:(def wheelEvent(self, event
    (Handle the wheel event (scrolling #
        :if self.chat_clicked_var
delta = event.angleDelta().y() / 120 # Normalize the delta
        ()mouse_pos = event.pos
    ) if delta > 0 and self.chat_box.is_mouse_on_chats_list(mouse_pos) and
self.chat_box_chats_index < 0): # or something
        Scrolling up #

self.chat_box_chats_index += 1

```



```

        ()self.update_chat_page_without_messages
    :(elif delta < 0 and self.chat_box.is_mouse_on_chats_list(mouse_pos
        Scrolling down, but prevent scrolling beyond the first message #
            self.chat_box_chats_index -= 1
        ()self.update_chat_page_without_messages
            :if self.social_clicked_var
                :try
                    delta = event.angleDelta().y() / 120 # Normalize the delta
                    ()mouse_pos = event.pos
                )if delta > 0 and self.friends_box.is_mouse_on_friends_box
                    mouse_pos) and self.friends_box_index_y_start >
                        :self.friends_box.default_starting_y
                            Scrolling up #

                                self.friends_box_index += 1
                                ()self.updated_social_page
                                    elif delta < 0 and
                                    self.friends_box.is_mouse_on_friends_box(mouse_pos) and
                                        :self.is_friends_box_full
                                            Scrolling down, but prevent scrolling beyond the first message #
                                                self.friends_box_index -= 1
                                                ()self.updated_social_page
                                                    :except Exception as e
                                                        ("){print(f"error in social_clicked scrolling error:{e

                                                            :(def hide_chat(self
                                                                (self.main_layout.addSpacerItem(self.spacer
                                                                    ()self.chat_box.hide

```

```

                                : (def show_chat(self
                                (self.main_layout.removeItem(self.spacer
                                (self.chat_box.setVisible(True

                                : (def updated_chat(self
                                (self.update_chat_page(True

                                : (def update_chat_page_without_messages(self
                                (self.update_chat_page(False

                                : (def update_chat_page(self, is_update_messages_box
                                : try
                                : if is_update_messages_box
                                self.is_messages_need_update = True
                                "" = text
                                : try
                                ()text = self.chat_box.text_entry.text
                                : except Exception as e
                                ("{print(f"error in updated chat1 {e
                                has_had_focus_of_search_bar =
                                ()self.chat_box.find_contact_text_entry.hasFocus
                                (self.stacked_widget.removeWidget(self.chat_box
                                self.chat_box.deleteLater() # Schedule deletion of the old ChatBox
                                                                widget
                                ()search_bar_text = self.chat_box.find_contact_text_entry.text
                                                                : try
                                self.chat_box = ChatBox(self.list_messages, parent=self,
                                                                (Network=self.Network
                                                                : except Exception as e

```

```

self.chat_box = ChatBox(self.list_messages, parent=self,
                        (Network=self.Network

("{print(f"error in creating chat_box on updated_chat_func : {e

        (self.stacked_widget.insertWidget(0, self.chat_box
            (self.chat_box.text_entry.setText(text
                :if len(search_bar_text) > 0
(self.chat_box.find_contact_text_entry.setText(search_bar_text
    (self.chat_box.find_contact_text_entry.setFocus(True
        ()self.chat_box.find_contact_text_entry.deselect
            :else
                :if has_had_focus_of_search_bar
(self.chat_box.find_contact_text_entry.setFocus(True
    ()self.chat_box.find_contact_text_entry.deselect
        :if self.chat_clicked_var
        (self.stacked_widget.setCurrentIndex(0
            :except Exception as e
self.chat_box = ChatBox(self.list_messages, parent=self,
                        (Network=self.Network

        ("{print(f"error in updated chat2 {e

            :(def update_values(self
self.friends_box.username = self.username

            :(def stop_all_media_player(self
        ()self.mp3_message_media_player.stop
            ()self.ringtone_media_player.stop
                ()self.calling_media_player.stop
                    ()self.sound_effect_media_player.stop

```

```
()self.playlist_media_player.stop
```

```
:(def closeEvent(self, event
```

```
This function is called when the window is closed #
```

```
()self.stop_all_media_player
```

```
()self.close_all_threads
```

```
()self.page_controller_object.quit_application
```

```
:(class SignUpPage(QWidget
```

```
:(def __init__(self, page_controller_object
```

```
()__super().__init
```

```
self.page_controller_object = page_controller_object
```

```
()self.init_ui
```

```
(self.password_not_match_label = QLabel('Password does not match', self
```

```
(";self.password_not_match_label.setStyleSheet("color: red; font-size: 12px
```

```
) = password_not_match_label_x, password_not_match_label_y
```

```
int(self.page_controller_object.screen_width * 0.44),
```

```
((int(self.page_controller_object.screen_height * 0.382
```

```
self.password_not_match_label.move(password_not_match_label_x,
```

```
(password_not_match_label_y
```

```
()self.password_not_match_label.hide
```

```
(x = int(self.page_controller_object.screen_width * 0.453
```

```
email_required_field_y = int(self.page_controller_object.screen_height *
```

```
(0.529
```

```
username_required_field_y = int(self.page_controller_object.screen_height
```

```
(* 0.307
```

```
password_required_field_y = int(self.page_controller_object.screen_height *  
                                (0.381
```

```
                                confirm_password_required_field_y =  
                                (int(self.page_controller_object.screen_height * 0.4555
```

```
(invalid_email_y = int(self.page_controller_object.screen_height * 0.529
```

```
password_too_short_y = int(self.page_controller_object.screen_height *  
                                (0.382
```

```
username_already_used_y = int(self.page_controller_object.screen_height *  
                                (0.307
```

```
self.email_required_field = self.create_label("Required field", (x,  
                                ((email_required_field_y
```

```
self.username_required_field = self.create_label("Required field", (x,  
                                ((username_required_field_y
```

```
self.password_required_field = self.create_label("Required field", (x,  
                                ((password_required_field_y
```

```
, "self.confirm_password_required_field = self.create_label("Required field  
((x, confirm_password_required_field_y)
```

```
((self.invalid_email = self.create_label("Invalid Email", (x, invalid_email_y
```

```
self.password_too_short = self.create_label("Password too short", (x,  
                                ((password_too_short_y
```

```
self.username_already_used = self.create_label("Username is taken", (x,  
                                ((username_already_used_y
```

```
()self.hide_every_error_label
```

```
:(def hide_every_error_label(self
```

```
()self.password_too_short.hide
```

```
()self.email_required_field.hide
```

```
()self.username_required_field.hide
```

```
()self.password_required_field.hide
```

```
()self.confirm_password_required_field.hide
```

```

        ()self.invalid_email.hide
    ()self.password_not_match_label.hide
    ()self.username_already_used.hide

    : (def create_label(self, text, position
        (label = QLabel(text, self
    (";label.setStyleSheet("color: red; font-size: 12px
    ([label.move(position[0], position[1
        ()label.hide
        return label

    : (def init_ui(self
        :try
        (label = QLabel("Create Account", self
            (username = QLineEdit(self
            (password = QLineEdit(self
            (password_confirm = QLineEdit(self
            (email = QLineEdit(self
            (password.setEchoMode(QLineEdit.Password
            (password_confirm.setEchoMode(QLineEdit.Password
            (image_button = QPushButton(self

        Load an image and set it as the button's icon #
        ("icon = QIcon("discord_app_assets/right-arrow-icon-27.png
        (image_button.setIcon(icon

    )width, height = int(self.page_controller_object.screen_width * 0.01), int
        (self.page_controller_object.screen_height * 0.018

```

```

        icon_size = QSize(width, height) # Set your desired size
        ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
(scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio

        (image_button.setIconSize(scaled_size

        image_button_x, image_button_y =
        ,((int(self.page_controller_object.screen_width * 0.471
int(self.page_controller_object.screen_height *
                                                                    ((0.217

        (image_button.move(image_button_x, image_button_y

        (image_button.clicked.connect(self.return_button_pressed

        Set placeholder text and color #
        ("username.setPlaceholderText("Username
        (";username.setStyleSheet("color: white

        ("password.setPlaceholderText("Password
        (";password.setStyleSheet("color: white

        ("password_confirm.setPlaceholderText("Confirm Password
        (";password_confirm.setStyleSheet("color: white

        ("email.setPlaceholderText("Email
        (";email.setStyleSheet("color: white

```

```

,(label_x, label_y = (int(self.page_controller_object.screen_width * 0.44
    ((int(self.page_controller_object.screen_height * 0.177
        username_x = label_x
    (username_y = int(self.page_controller_object.screen_height * 0.259
        password_x = label_x
    (password_y = int(self.page_controller_object.screen_height * 0.333
        password_confirm_x = label_x
password_confirm_y = int(self.page_controller_object.screen_height *
                                                                    (0.407
    (email_y = int(self.page_controller_object.screen_height * 0.481
        email_x = label_x
        (label.move(label_x, label_y
    (username.move(username_x, username_y
        (password.move(password_x, password_y
    (password_confirm.move(password_confirm_x, password_confirm_y
        (email.move(email_x, email_y

Create button #
    (submit_button = QPushButton('Submit', self
        )submit_button.clicked.connect
    lambda: self.submit_form(username.text(), password.text(),
        (((password_confirm.text(), email.text

        submit_button_x, submit_button_y =
    ,((int(self.page_controller_object.screen_width * 0.427
int(self.page_controller_object.screen_height *
                                                                    ((0.555

    (submit_button.move(submit_button_x, submit_button_y

```



```

        ""submit_button.setStyleSheet
            } QPushButton
        ;background-color: #6fa8b6
            ;color: #f0f1f1
/* padding: 10px; /* Adjust the padding to make the button smaller
            ;border: 1px solid #2980b9
            ;border-radius: 5px
min-width: 200px; /* Adjust the min-width to set the minimum width
                        /* of the button
        max-width: 300px; /* Adjust the max-width to set the maximum
                        /* width of the button
min-height: 30px; /* Adjust the min-height to set the minimum height
                        /* of the button
        max-height: 60px; /* Adjust the max-height to set the maximum
                        /* height of the button
            /* font-size: 16px; /* Set your desired font size
margin-top: 10px; /* Adjust the margin-top to set the top margin of
                        /* the button
            {
                } QPushButton:hover
            ;background-color: #2980b9
            {
                } QPushButton:pressed
            ;background-color: #1f618d
            {
                ("
                Set styles #
            """)self.setStyleSheet
            } QWidget
/* background-color: #141c4b; /* Set your desired background color

```

```

        {
            } QLabel
            ;color: #f0f1f1
            ;font-size: 20px
            ;margin-bottom: 20px
        {
            } QLineEdit
;background-color: #6fa8b6
            ;color: #f0f1f1
            ;padding: 10px
;border: 1px solid #2980b9
            ;border-radius: 5px
            ;font-size: 16px
            ;margin-bottom: 10px
        {

            ("""
                :except Exception as e
            ({print(f"error in Sign_up_page {e

:(def submit_form(self, username, password, password_confirm, email
            n = self.page_controller_object.n
            ()self.hide_every_error_label
            is_info_valid = True
== if username == "" or password == "" or password_confirm == "" or email
            :""

            is_info_valid = False
            :"" == if email
            ()self.email_Required_field.show

```

```

        :"" == if password
        ()self.password_Required_field.show
        :"" == if password_confirm
        ()self.confirm_password_Required_field.show
        :"" == if username
        ()self.username_Required_field.show

if password != password_confirm and password != "" and password_confirm
        :"" !=
        is_info_valid = False
        ()self.password_not_match_label.show
:elif password == password_confirm and len(password) < 8
        is_info_valid = False
        ()self.password_too_short.show
        :"" != if not is_email_valid(email) and email
        is_info_valid = False
        ()self.invalid_email.show
        :if is_info_valid
        (n.send_sign_up_info(username, password, email
        ()data = n.recv_str
        ("message_type = data.get("message_type
        :if message_type == "code
        ("action, destination = data.get("action"), data.get("sent_to
        :if action == "sent" and destination == "email
        ()self.page_controller_object.change_to_verification_code_page
        :elif message_type == "sign_up
        ("result = data.get("sign_up_status
        :if result == "invalid
        ()self.username_already_used.show

```

```

        : (def return_button_pressed(self
        ()self.page_controller_object.change_to_login_page

class LoginPage(QWidget
    : (def __init__(self, page_controller_object
        ()__super().__init
        self.page_controller_object = page_controller_object
        ()self.init_ui
        (self.visibility_password_button = QPushButton(self
            Load an image and set it as the button's icon #
            self.show_password_icon =
            ("QIcon("discord_app_assets/show_password_icon.png
            self.hide_password_icon =
            ("QIcon("discord_app_assets/hide_password_icon1.png
        (self.visibility_password_button.setIcon(self.show_password_icon
        "self.current_icon = "discord_app_assets/show_password_icon.png

width, height = int(self.page_controller_object.screen_width * 0.02), int
        (self.page_controller_object.screen_height * 0.036

        icon_size = QSize(width, height) # Set your desired size
        icon_actual_size =
self.show_password_icon.actualSize(self.show_password_icon.availableSizes()[
        ([0

        (scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio
        (self.visibility_password_button.setIconSize(scaled_size

        ) = visibility_password_button_x, visibility_password_button_y

```

```

        int(self.page_controller_object.screen_width * 0.528),
        ((int(self.page_controller_object.screen_height * 0.333

self.visibility_password_button.move(visibility_password_button_x,
                                     (visibility_password_button_y

self.visibility_password_button.clicked.connect(self.show_password_button_pres
                                              (sed

```

```

        """)self.visibility_password_button.setStyleSheet
                } QPushButton
                ;background-color: #6fa8b6
                ;background-repeat: no-repeat
                ;background-position: center
                {
                (""
                (self.password = QLineEdit(self
                (self.password.setEchoMode(QLineEdit.Password
                ("self.password.setPlaceholderText("Password
                (";self.password.setStyleSheet("color: white

                ) = password_x, password_y
                int(self.page_controller_object.screen_width * 0.44),
                ((int(self.page_controller_object.screen_height * 0.333

                (self.password.move(password_x, password_y

                (self.username_entry = QLineEdit(self
                self.remember_me_status = False
                ("self.username_entry.setPlaceholderText("Username

```

```

        (";self.username_entry.setStyleSheet("color: white

        ) = username_entry_x, username_entry_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.259

        (self.username_entry.move(username_entry_x, username_entry_y
        (self.incorrect_label = QLabel('Username or Password incorrect', self
        )self.incorrect_label.setStyleSheet
        color: red; font-size: 12px;") # Set the text color to blue and font size to"
        12px

        ) = incorrect_label_x, incorrect_label_y
        int(self.page_controller_object.screen_width * 0.445),
        ((int(self.page_controller_object.screen_height * 0.308

        (self.incorrect_label.move(incorrect_label_x, incorrect_label_y
        ()self.incorrect_label.hide

        (self.user_is_logged_in = QLabel('Username already logged in', self
        )self.user_is_logged_in.setStyleSheet
        color: red; font-size: 12px;") # Set the text color to blue and font size to"
        12px

        ) = user_is_logged_in_x, user_is_logged_in_y
        int(self.page_controller_object.screen_width * 0.445),
        ((int(self.page_controller_object.screen_height * 0.308

        (self.user_is_logged_in.move(user_is_logged_in_x, user_is_logged_in_y
        ()self.user_is_logged_in.hide

```

```

:(def init_ui(self

    (label = QLabel("Welcome Back", self
        (image_button = QPushButton(self

        Load an image and set it as the button's icon #
        ("icon = QIcon("discord_app_assets/right-arrow-icon-27.png
            (image_button.setIcon(icon
        )width, height = int(self.page_controller_object.screen_width * 0.01), int
            (self.page_controller_object.screen_height * 0.018

        icon_size = QSize(width, height) # Set your desired size
        ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
        (scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio

        (image_button.setIconSize(scaled_size

        ) = image_button_x, image_button_y
        int(self.page_controller_object.screen_width * 0.471),
        ((int(self.page_controller_object.screen_height * 0.217

        (image_button.move(image_button_x, image_button_y

        Create "Forgot your password?" label #
        (forgot_password_label = QPushButton('Forgot your password?', self
            (make_q_object_clear(forgot_password_label
                )forgot_password_label.setStyleSheet

```

```
color: white; font-size: 12px;") # Set the text color to blue and font size"
                                                    to 12px
```

```
(sign_up_label = QPushButton("Don't have a user yet? sign up here", self
                               (make_q_object_clear(sign_up_label
                                                       )sign_up_label.setStyleSheet
```

```
color: white; font-size: 12px;") # Set the text color to blue and font size"
                                                    to 12px
```

```
(sign_up_label.clicked.connect(self.move_to_sign_up_page

                               ) = sign_up_label_x, sign_up_label_y
```

```
int(self.page_controller_object.screen_width * 0.4244),
    ((int(self.page_controller_object.screen_height * 0.495
```

```
(sign_up_label.move(sign_up_label_x, sign_up_label_y
```

```
(checkbox = QCheckBox('Keep me signed in', self
```

```
("{checkbox.setStyleSheet("QCheckBox { color: white; font-size: 12px
(checkbox.stateChanged.connect(self.on_checkbox_change
```

```
) = checkbox_x, checkbox_y
```

```
int(self.page_controller_object.screen_width * 0.445),
    ((int(self.page_controller_object.screen_height * 0.3842
```

```
(checkbox.move(checkbox_x, checkbox_y
```

```
Connect the linkActivated signal to a custom slot #
```

```
(forgot_password_label.clicked.connect(self.forgot_password_clicked
```

```
) = forgot_password_label_x, forgot_password_label_y
```



```

        int(self.page_controller_object.screen_width * 0.445),
        ((int(self.page_controller_object.screen_height * 0.412

forgot_password_label.move(forgot_password_label_x,
                            (forgot_password_label_y

                                ) = label_x, label_y

int(self.page_controller_object.screen_width * 0.445),
    ((int(self.page_controller_object.screen_height * 0.177

        (label.move(label_x, label_y

                                Create button #

        (submit_button = QPushButton('Login', self
        (submit_button.clicked.connect(self.submit_form

                                ) = submit_button_x, submit_button_y

int(self.page_controller_object.screen_width * 0.424),
    ((int(self.page_controller_object.screen_height * 0.43

(submit_button.move(submit_button_x, submit_button_y
                    ""))submit_button.setStyleSheet
                                } QPushButton
                                ;background-color: #6fa8b6
                                ;color: #f0f1f1

/* padding: 10px; /* Adjust the padding to make the button smaller
                                ;border: 1px solid #2980b9
                                ;border-radius: 5px

min-width: 200px; /* Adjust the min-width to set the minimum width of
                                /* the button

```

```

max-width: 300px; /* Adjust the max-width to set the maximum width
/* of the button
min-height: 30px; /* Adjust the min-height to set the minimum height of
/* the button
max-height: 60px; /* Adjust the max-height to set the maximum height
/* of the button
/* font-size: 16px; /* Set your desired font size
margin-top: 10px; /* Adjust the margin-top to set the top margin of the
/* button
{
} QPushButton:hover
;background-color: #2980b9
{
} QPushButton:pressed
;background-color: #1f618d
{
("

Set styles #
""")self.setStyleSheet
} QWidget

/* background-color: #141c4b; /* Set your desired background color
{
} QLabel
;color: #f0f1f1
;font-size: 20px
;margin-bottom: 20px
{
} QLineEdit
;background-color: #6fa8b6

```

```

;color: #f0f1f1
;padding: 10px
;border: 1px solid #2980b9
;border-radius: 5px
;font-size: 16px
;margin-bottom: 10px
{
  (""

```

```

:(def on_checkbox_change(self, state
This function is called when the checkbox state changes #
if state == 2: # 2 corresponds to checked state
    self.remember_me_status = True
    :else
    self.remember_me_status = False

```

```

:(def submit_form(self
    :try
        n = self.page_controller_object.n
        ()self.incorrect_label.hide
        ()self.user_is_logged_in.hide
        ()self.username = self.username_entry.text
        ()password = self.password.text
        (n.send_login_info(self.username, password
        ()data = n.recv_str
        ("message_type = data.get("message_type
        :if message_type == "login
        ("login_status = data.get("login_status

```

```

        :if login_status == "confirm
        ("print("logged in successfully
        ()n.connect_between_udp_port_address_to_username
        ()self.hide
        :if self.remember_me_status
        ()n.ask_for_security_token
        ("print("You will be remembered
self.page_controller_object.main_page.username = self.username
        ()self.page_controller_object.main_page.update_values
        self.page_controller_object.is_logged_in = True
        ()self.page_controller_object.start_receive_thread_after_login
        ()self.page_controller_object.main_page.start_listen_udp_thread
        ()self.page_controller_object.change_to_splash_page
        :elif login_status == "already_logged_in
        ("print("User logged in from another device
        ()self.user_is_logged_in.show
        :elif login_status == "2fa
        ("print("You have 2fa On
        self.page_controller_object.is_waiting_for_2fa_code = True
        ()self.page_controller_object.change_to_verification_code_page
        :else
        ("print("login info isn't correct
        ()self.incorrect_label.show
        :except Exception as e
        ("print(f"error in trying to login {e

        :(def forgot_password_clicked(self
        ()self.page_controller_object.change_to_forget_password_page

```

```

        : (def move_to_sign_up_page(self
                                                    : try
        ()self.page_controller_object.change_to_sign_up_page
                                                    : except Exception as e
                                                    (print(e

        : (def show_password_button_pressed(self
: "if self.current_icon == "discord_app_assets/show_password_icon.png
        (self.password.setEchoMode(QLineEdit.Normal
"self.current_icon = "discord_app_assets/hide_password_icon.png
        (self.visibility_password_button.setIcon(self.hide_password_icon
                                                    : else
        (self.password.setEchoMode(QLineEdit.Password
"self.current_icon = "discord_app_assets/show_password_icon.png
        (self.visibility_password_button.setIcon(self.show_password_icon

        : (class VideoClient(QMainWindow
        : (def __init__(self, main_page
            ()__super().__init
            self.main_page = main_page
            ()self.init_ui

        : (def init_ui(self
            (self.central_widget = QWidget(self
            (self.setCentralWidget(self.central_widget

```

```

        (self.image_label = QLabel(self
            (self.image_label.move(0, 0

(layout = QVBoxLayout(self.central_widget
    (layout.addWidget(self.image_label

    x, y = (int(self.main_page.screen_width * 0.052),
            ((int(self.main_page.screen_height * 0.092
width, height = (int(self.main_page.screen_width * 0.416),
                ((int(self.main_page.screen_height * 0.555
            (self.setGeometry(x, y, width, height
            ('self.setWindowTitle('Video Client

        : (def display_frame(self, frame
            height, width, channel = frame.shape
                bytes_per_line = 3 * width
        ()screen_size = QApplication.primaryScreen().size
            ()screen_width = screen_size.width
            ()screen_height = screen_size.height

    q_image = QImage(frame.data, width, height, bytes_per_line,
        ()QImage.Format_RGB888).rgbSwapped
    pixmap = QPixmap.fromImage(q_image).scaled(screen_width,
                                                (screen_height

        Clear the existing pixmap #
            ()self.image_label.clear

        Set the new pixmap #
            (self.image_label.setPixmap(pixmap

```

```

        : (def keyPressEvent(self, event
            : if event.key() == Qt.Key_Escape
                ()self.main_page.showMaximized
            ()self.main_page.Network.stop_watching_current_stream
            ("print(f'stopped watching {self.main_page.watching_user} share screen
                self.main_page.is_watching_screen = False
                "" = self.main_page.watching_user
            self.main_page.watching_type = None
                ()self.close

```

```

        : (class ForgetPasswordPage(QWidget
            : (def __init__(self, page_controller_object
                ()__super().__init
            self.page_controller_object = page_controller_object
                ()self.init_ui
            (self.username = QLineEdit(self
                ) = username_x, username_y
            int(self.page_controller_object.screen_width * 0.44),
                ((int(self.page_controller_object.screen_height * 0.26

            (self.username.move(username_x, username_y
            ("self.username.setPlaceholderText("Username
                (";self.username.setStyleSheet("color: white
                    (self.email = QLineEdit(self
                        ("self.email.setPlaceholderText("Email
                            (";self.email.setStyleSheet("color: white
                                ) = email_x, email_y

```

```

int(self.page_controller_object.screen_width * 0.44),
    ((int(self.page_controller_object.screen_height * 0.33

        (self.email.move(email_x, email_y

        :(def create_label(self, text, position
            (label = QLabel(text, self
            (";label.setStyleSheet("color: red; font-size: 12px
            ([label.move(position[0], position[1
                ()label.hide
                return label

        :(def init_ui(self
            (label = QLabel("Forget password", self

            (image_button = QPushButton(self
                Load an image and set it as the button's icon #
            ("icon = QIcon("discord_app_assets/right-arrow-icon-27.png
                (image_button.setIcon(icon
            )width, height = int(self.page_controller_object.screen_width * 0.01), int
            (self.page_controller_object.screen_height * 0.018

            icon_size = QSize(width, height) # Set your desired size
            ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
            (scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio
                (image_button.setIconSize(scaled_size
                    ) = code_label_x, code_label_y
            int(self.page_controller_object.screen_width * 0.471),
            ((int(self.page_controller_object.screen_height * 0.2175

```



```
(image_button.move(code_label_x, code_label_y  
(image_button.clicked.connect(self.return_button_pressed
```

Set placeholder text and color #

```
Create "Forgot your password?" label #  
(code_label = QLabel('<a href="forgot">Resend code</a>', self  
(code_label.setTextInteractionFlags(Qt.TextBrowserInteraction  
code_label.setOpenExternalLinks(False) # Disable external links to capture  
linkActivated signal  
code_label.setStyleSheet("color: blue; font-size: 15px;") # Set the text color  
to blue and font size to 12px
```

```
Connect the linkActivated signal to a custom slot #  
(code_label.linkActivated.connect(self.resend_code_clicked  
)= code_label_x, code_label_y  
int(self.page_controller_object.screen_width * 0.45),  
((int(self.page_controller_object.screen_height * 0.388
```

```
(code_label.move(code_label_x, code_label_y  
)= label_x, label_y  
int(self.page_controller_object.screen_width * 0.44),  
((int(self.page_controller_object.screen_height * 0.177
```

```
(label.move(label_x, label_y
```

Create button #

```
(submit_button = QPushButton('Submit info', self  
(submit_button.clicked.connect(self.submit_form
```

```

        ) = submit_button_x, submit_button_y
        int(self.page_controller_object.screen_width * 0.424),
        ((int(self.page_controller_object.screen_height * 0.416

(submit_button.move(submit_button_x, submit_button_y

        ""))submit_button.setStyleSheet

            } QPushButton

            ;background-color: #6fa8b6

            ;color: #f0f1f1

/* padding: 10px; /* Adjust the padding to make the button smaller

            ;border: 1px solid #2980b9

            ;border-radius: 5px

min-width: 200px; /* Adjust the min-width to set the minimum width of
/* the button

max-width: 300px; /* Adjust the max-width to set the maximum width
/* of the button

min-height: 30px; /* Adjust the min-height to set the minimum height of
/* the button

max-height: 60px; /* Adjust the max-height to set the maximum height
/* of the button

/* font-size: 16px; /* Set your desired font size

margin-top: 10px; /* Adjust the margin-top to set the top margin of the
/* button

        {

            } QPushButton:hover

            ;background-color: #2980b9

            {

            } QPushButton:pressed

            ;background-color: #1f618d

            {

```

```

        } QPushButton:hover
;background-color: #2980b9
    {
        } QPushButton:pressed
;background-color: #1f618d
    {
        ("

Set styles #
""")self.setStyleSheet
    } QWidget
/* background-color: #141c4b; /* Set your desired background color
    {
        } QLabel
;color: #f0f1f1
;font-size: 20px
;margin-bottom: 20px
    {
        } QLineEdit
;background-color: #6fa8b6
;color: #f0f1f1
;padding: 10px
;border: 1px solid #2980b9
;border-radius: 5px
;font-size: 16px
;margin-bottom: 10px
    {
        ("

```

```

        : (def return_button_pressed(self
()self.page_controller_object.change_to_login_page

        : (def submit_form(self
n = self.page_controller_object.n
()username = self.username.text
()email = self.email.text
:if is_email_valid(email) and len(username) > 0
(n.send_username_and_email_forget_password(username, None, email
()data = n.recv_str
("message_type = data.get("message_type
:"if message_type == "forget_password
("result = data.get("forget_password_status
:"if result == "valid
("print("Server send code to email
()self.page_controller_object.change_to_verification_code_page
:"elif result == "invalid
pass

        : (def resend_code_clicked(self
(print(4

        : (class VerificationCodePage(QWidget
: (def __init__(self, page_controller_object
()__super().__init
self.page_controller_object = page_controller_object

```

```

        ()self.init_ui

        (self.info_label = QLabel(self
    pixmap = QPixmap('discord_app_assets/info_icon.png') # Replace with the
        path to your 'i' icon
        ) = width, height

    int(self.page_controller_object.screen_width * 0.02),
        ((int(self.page_controller_object.screen_height * 0.037
    (scaled_pixmap = pixmap.scaled(width, height, Qt.KeepAspectRatio
        (self.info_label.setPixmap(scaled_pixmap
        (self.info_label.setAlignment(Qt.AlignCenter
    self.info_label.setStyleSheet("background-color: #141c4b; border-radius:
        (";25px
    ('self.info_label.setToolTip('A mail will be sent to your chosen email address
        ) = info_label_x, info_label_y

    int(self.page_controller_object.screen_width * 0.468),
        ((int(self.page_controller_object.screen_height * 0.231

    (self.info_label.move(info_label_x, info_label_y
        (self.info_label.setMouseTracking(True
        (self.info_label.installEventFilter(self
            (self.code = QLineEdit(self
            (self.code.setMaxLength(6
            (())self.code.setValidator(QIntValidator

    Set placeholder text and color #

    ("self.code.setPlaceholderText("code
    (";self.code.setStyleSheet("color: white
        ) = code_x, code_y

```

```

        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.296

                                (self.code.move(code_x, code_y

                                ) = successfully_signed_up_x, successfully_signed_up_y
                                int(self.page_controller_object.screen_width * 0.4427),
                                ((int(self.page_controller_object.screen_height * 0.5

, "self.successfully_signed_up = self.create_label("successfully signed up
                                successfully_signed_up_x,)
                                ((successfully_signed_up_y
                                ()self.successfully_signed_up.hide

                                (self.image_button = QPushButton(self
                                Load an image and set it as the button's icon #
                                ("icon = QIcon("discord_app_assets/right-arrow-icon-27.png
                                (self.image_button.setIcon(icon
                                )width, height = int(self.page_controller_object.screen_width * 0.01), int
                                (self.page_controller_object.screen_height * 0.018
                                icon_size = QSize(width, height) # Set your desired size
                                ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
                                (scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio
                                (self.image_button.setIconSize(scaled_size

                                ) = image_button_x, image_button_y
                                int(self.page_controller_object.screen_width * 0.445),
                                ((int(self.page_controller_object.screen_height * 0.19
                                (self.image_button.move(image_button_x, image_button_y
                                ()self.image_button.hide

```

```

(self.image_button.clicked.connect(self.return_button_pressed
        """)self.setStyleSheet
        } QWidget
/* background-color: #141c4b; /* Set your desired background color
        {
        } QLabel
        ;color: #f0f1f1
        ;font-size: 20px
        ;margin-bottom: 20px
        {
        } QLineEdit
;background-color: #6fa8b6
        ;color: #f0f1f1
        ;padding: 10px
;border: 1px solid #2980b9
        ;border-radius: 5px
        ;font-size: 16px
        ;margin-bottom: 10px
        {
        } QPushButton:hover
;background-color: #2980b9
        {
        } QPushButton:pressed
;background-color: #1f618d
        {
        (""

:(def return_button_pressed(self

```

```

        ()self.page_controller_object.change_to_login_page

    def eventFilter(self, obj, event):
        try:
            if obj == self.info_label and event.type() == event.Enter:
                Set a fixed position for the tooltip #
                (xpos = int(self.page_controller_object.screen_width * 0.416
                (ypos = int(self.page_controller_object.screen_height * 0.277
                (fixed_position = QPoint(xpos, ypos
                ((QToolTip.showText(fixed_position, self.info_label.toolTip
            return True # Consume the event to prevent further processing
        except Exception as e:
            print(f"error in eventFilter {e}")

    def create_label(self, text, position):
        label = QLabel(text, self)
        label.setStyleSheet("color: green; font-size: 12px")
        label.move(position[0], position[1])
        label.hide()
        return label

    def init_ui(self):
        label = QLabel("Verification code", self)
        code_label = QLabel('<a href="resend">Resend code</a>', self)
        code_label.setTextInteractionFlags(Qt.TextBrowserInteraction)
        code_label.setOpenExternalLinks(False) # Disable external links to capture
                                                linkActivated signal

```



```
code_label.setStyleSheet("color: blue; font-size: 15px;") # Set the text color
                                                    to blue and font size to 12px
```

```

        Connect the linkActivated signal to a custom slot #
        (code_label.linkActivated.connect(self.resend_code_clicked
                                                    ) = code_label_x, code_label_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.3796

```

```

        (code_label.move(code_label_x, code_label_y
                                                    ) = label_x, label_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.166

```

```
        (label.move(label_x, label_y
```

```
        Create button #
```

```

        (submit_button = QPushButton('Submit code', self
        (submit_button.clicked.connect(self.submit_form
                                                    ) = submit_button_x, submit_button_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.416

```

```
        (submit_button.move(submit_button_x, submit_button_y
```

```
        Set styles #
```

```
        """)self.setStyleSheet
```

```
        } QWidget
```

```
/* background-color: #141c4b; /* Set your desired background color
```

```
{
```

```

        } QLabel
        ;color: #f0f1f1
        ;font-size: 20px
        ;margin-bottom: 20px
    {
        } QLineEdit
        ;background-color: #6fa8b6
        ;color: #f0f1f1
        ;padding: 10px
        ;border: 1px solid #2980b9
        ;border-radius: 5px
        ;font-size: 16px
        ;margin-bottom: 10px
    {
        } QPushButton
        ;background-color: #6fa8b6
        ;color: #f0f1f1
        ;padding: 20px
        ;border: 1px solid #2980b9
        ;border-radius: 5px
        ;min-width: 30px
        ;max-width: 100px
        /* min-height: 0px; /* Set the minimum height to 50 pixels
        /* max-height: 16px; /* Set the maximum height to 200 pixels
        /* font-size: 16px; /* Set your desired font size
        ;margin-top: 20px
    {
        } QPushButton:hover

```

```

;background-color: #2980b9
{
    } QPushButton:pressed
;background-color: #1f618d
{
    (""

:(def submit_form(self
n = self.page_controller_object.n
()code = self.code.text
:try
:if len(code) == 6
:if not self.page_controller_object.is_waiting_for_2fa_code
(n.send_sign_up_verification_code(code
:else
(n.send_login_2fa_code(code
("print("Sent verification code to server
()data = n.recv_str
("message_type = data.get("message_type
:"if message_type == "sign_up
("kind = data.get("action
:"if kind == "code
("result = data.get("code_status
:"if result == "valid
("print("Server got the code
()self.successfully_signed_up.show
()self.image_button.show
:"elif result == "invalid

```

```

        pass
    elif message_type == "forget_password
        ("kind = data.get("action
        ("result = data.get("code_status
            :if kind == "code
            :if result == "valid

    ()self.page_controller_object.change_to_change_password_page
        :elif result == "invalid
        pass
        :elif message_type == "2fa
        :try
            ("kind = data.get("action
            ("result = data.get("code_status
                :if kind == "code
                :if result == "valid

            ("print("logged in successfully
    ()n.connect_between_udp_port_address_to_username
        ()self.hide
        if
        :self.page_controller_object.login_page.remember_me_status
        ()n.ask_for_security_token
        ("print("You will be remembered
        :try

    self.page_controller_object.main_page.username =
        ()self.page_controller_object.login_page.username_entry.text
    ()self.page_controller_object.main_page.update_values
    self.page_controller_object.is_logged_in = True

    ()self.page_controller_object.start_receive_thread_after_login

```

```

        ()self.page_controller_object.main_page.start_listen_udp_thread
()self.page_controller_object.change_to_splash_page
        :except Exception as e
        ("{print(f"error in 2fa {e
            :elif result == "invalid
                pass
            :except Exception as e
            ("{print(f"error in 2fa {e
                :except Exception as e
            ("{print(f"error in submit_form verification code {e

        :(def resend_code_clicked(self, link
            :if link == "resend
                ("print("resend code

        :(class ChangePasswordPage(QWidget
        :(def __init__(self, page_controller_object
            ()__super().__init
            self.page_controller_object = page_controller_object
            ()self.init_ui
        (self.change_password_label = QLabel("Change password:", self
            ) = change_password_label_x, change_password_label_y
                int(self.page_controller_object.screen_width * 0.44),
                ((int(self.page_controller_object.screen_height * 0.185
        self.change_password_label.move(change_password_label_x,
            (change_password_label_y
        (self.new_password = QLineEdit(self

```

```

        ) = new_password_x, new_password_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.26

(self.new_password.move(new_password_x, new_password_y
    ("self.new_password.setPlaceholderText("New password
        (";self.new_password.setStyleSheet("color: white
            self.status = False
            (self.image_button = QPushButton(self
                Load an image and set it as the button's icon #
                ("icon = QIcon("discord_app_assets/right-arrow-icon-27.png
                    (self.image_button.setIcon(icon
)width, height = int(self.page_controller_object.screen_width * 0.01), int
    (self.page_controller_object.screen_height * 0.018

        icon_size = QSize(width, height) # Set your desired size
        ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
(scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio
    (self.image_button.setIconSize(scaled_size
        ) = image_button_x, image_button_y
        int(self.page_controller_object.screen_width * 0.47),
        ((int(self.page_controller_object.screen_height * 0.19

(self.image_button.move(image_button_x, image_button_y

(self.image_button.clicked.connect(self.return_button_pressed
    ()self.image_button.hide
        """)self.setStyleSheet
    } QWidget

```

```

/* background-color: #141c4b; /* Set your desired background color

                                {
                                } QLabel
                                ;color: #f0f1f1
                                ;font-size: 20px
                                ;margin-bottom: 20px
                                {
                                } QLineEdit
;background-color: #6fa8b6
                                ;color: #f0f1f1
                                ;padding: 10px
;border: 1px solid #2980b9
                                ;border-radius: 5px
                                ;font-size: 16px
                                ;margin-bottom: 10px
                                {
                                } QPushButton:hover
;background-color: #2980b9
                                {
                                } QPushButton:pressed
;background-color: #1f618d
                                {
                                }

                                : (def return_button_pressed(self
                                ()self.page_controller_object.change_to_login_page

                                : (def create_label(self, text, position

```

```

        (label = QLabel(text, self
";label.setStyleSheet("color: red; font-size: 12px
        ([label.move(position[0], position[1
        )label.hide
        return label

:(def init_ui(self
(self.too_short = QLabel("Password too short", self

        (image_button = QPushButton(self
        Load an image and set it as the button's icon #
(icon = QIcon("discord_app_assets/right-arrow-icon-27.png
        (image_button.setIcon(icon
)width, height = int(self.page_controller_object.screen_width * 0.01), int
        (self.page_controller_object.screen_height * 0.018
        icon_size = QSize(width, height) # Set your desired size
([icon_actual_size = icon.actualSize(icon.availableSizes())[0
(scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio
        (image_button.setIconSize(scaled_size
        ) = image_button_x, image_button_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.217
        (image_button.move(image_button_x, image_button_y
        (image_button.clicked.connect(self.return_button_pressed

        self.was_password_changed = False
        ) = too_short_x, too_short_y
        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.315

```



```

        (self.too_short.move(too_short_x, too_short_y
                                )self.too_short.hide
                                )self.too_short.setStyleSheet
color: red; font-size: 14px;") # Set the text color to blue and font size to"
12px

```

```

self.password_already_changed = QLabel("Password already changed",
                                         (self
                                         ) = password_already_changed_x, password_already_changed_y
                                         int(self.page_controller_object.screen_width * 0.44),
                                         ((int(self.page_controller_object.screen_height * 0.314
                                         self.password_already_changed.move(password_already_changed_x,
                                         (password_already_changed_y
                                         )self.password_already_changed.hide
                                         )self.password_already_changed.setStyleSheet
color: red; font-size: 14px;") # Set the text color to blue and font size to"
12px

```

```

(self.changed_password_label = QLabel("Password changed", self
    ) = changed_password_label_x, changed_password_label_y
    int(self.page_controller_object.screen_width * 0.44),
    ((int(self.page_controller_object.screen_height * 0.315
self.changed_password_label.move(changed_password_label_x,
    (changed_password_label_y
    )self.changed_password_label.hide
    Create button #
    (submit_button = QPushButton('Submit info', self
    (submit_button.clicked.connect(self.submit_form
    ) = submit_button_x, submit_button_y

```

```

        int(self.page_controller_object.screen_width * 0.44),
        ((int(self.page_controller_object.screen_height * 0.415
(submit_button.move(submit_button_x, submit_button_y
                    ""))submit_button.setStyleSheet
                        } QPushButton
                    ;background-color: #6fa8b6
                        ;color: #f0f1f1
/* padding: 10px; /* Adjust the padding to make the button smaller
                    ;border: 1px solid #2980b9
                        ;border-radius: 5px
min-width: 200px; /* Adjust the min-width to set the minimum width of
                        /* the button
max-width: 300px; /* Adjust the max-width to set the maximum width
                        /* of the button
min-height: 30px; /* Adjust the min-height to set the minimum height of
                        /* the button
max-height: 60px; /* Adjust the max-height to set the maximum height
                        /* of the button

/* font-size: 16px; /* Set your desired font size
margin-top: 10px; /* Adjust the margin-top to set the top margin of the
                        /* button
                        {
                            } QPushButton:hover
                    ;background-color: #2980b9
                        {
                            } QPushButton:pressed
                    ;background-color: #1f618d
                        {
                            } QPushButton:pressed
                    ;background-color: #2980b9

```

```

        {
            } QPushButton:pressed
;background-color: #1f618d
        {
            (""

Set styles #
        """)self.setStyleSheet
        } QWidget
/* background-color: #141c4b; /* Set your desired background color
        {
            } QLabel
;color: #f0f1f1
;font-size: 20px
;margin-bottom: 20px
        {
            } QLineEdit
;background-color: #6fa8b6
;color: #f0f1f1
;padding: 10px
;border: 1px solid #2980b9
;border-radius: 5px
;font-size: 16px
;margin-bottom: 10px
        {
            (""

:(def submit_form(self

```

```

        n = self.page_controller_object.n
        ()self.too_short.hide
        ()self.password_already_changed.hide
        ()self.changed_password_label.hide
:if len(self.new_password.text()) >= 8 and not self.was_password_changed
        ((n.send_new_password(self.new_password.text
            ("print("Password changed
        ()self.changed_password_label.show
            self.status = True
        self.was_password_changed = True
        :elif self.was_password_changed
        ()self.password_already_changed.show
            :else
            ()self.too_short.show

        :(def resend_code_clicked(self
            (print(4

:(class ServerIsDownPage(QWidget
:(def __init__(self, page_controller_object
        :try
            ()__super().__init
        self.page_controller_object = page_controller_object

        server_is_offline_x, server_is_offline_y =
            ,((int(self.page_controller_object.screen_width * 0.44
int(self.page_controller_object.screen_height
            ((* 0.185

```

```

(self.server_is_offline = QLabel("Server is Offline...", self

(self.server_is_offline.move(server_is_offline_x, server_is_offline_y
    ("self.server_is_offline.setStyleSheet("color: white; font-size:20px
        """)self.setStyleSheet
            } QWidget
/* background-color: #141c4b; /* Set your desired background color
        {
        (""
        :except Exception as e
        ("print("error in server id down page

:class PageController
    :(def __init__(self
        ()self.n = ClientNet
        ()is_connected = self.n.connect_tcp
        ()self.screen_width, self.screen_height = pyautogui.size
        (self.app = QApplication(sys.argv

        :if is_connected
            :try
                self.receive_thread_after_login =
                ((=threading.Thread(target=self.thread_recv_messages, args
                    self.is_logged_in = False
                    self.is_waiting_for_2fa_code = False
                (self.splash_page = SplashScreen(self
                    ()self.splash_page.showMaximized
                (self.sign_up_page = SignUpPage(self

```

```

(self.forget_password_page = ForgetPasswordPage(self
    (self.login_page = LoginPage(self
        (self.main_page = MainPage(self.n, self
(self.change_password_page = ChangePasswordPage(self
    (self.verification_code_page = VerificationCodePage(self
        ()self.main_page.showMaximized
            ()self.main_page.hide
                ()self.sign_up_page.showMaximized
                    ()self.forget_password_page.showMaximized
                        ()self.login_page.showMaximized
                            ()self.login_page.hide
                                ()self.verification_code_page.showMaximized
                                    ()self.sign_up_page.hide
                                        ()self.forget_password_page.hide
                                            ()self.verification_code_page.hide
                                                ()self.change_password_page.showMaximized
                                                    ()self.change_password_page.hide
                                                        self.current_page = self.login_page
                                                            :except Exception as e
                                                                (print(e
                                                                    :else
(self.server_is_down_page = ServerIsDownPage(self
    ()self.server_is_down_page.showMaximized
        ()_self.app.exec

:(def start_receive_thread_after_login(self
    ()self.receive_thread_after_login.start

```

```

        : (def quit_application(self
            : if self.is_logged_in
            ("...print("closing app
            self.is_logged_in = False
            ()self.n.close
            ()self.main_page.close_all_threads
            ()self.close_all_pages
            del self.app
            ()sys.exit

        : (def log_out(self
            : try
            ("print("logging out
            ()self.n.send_logout_message
            ()self.main_page.close_all_threads
            self.current_page = None
            self.is_logged_in = False
            ()self.close_all_pages
            ()self.clear_all_pages
            ()self.hide_all_pages
            ()self.receive_thread_after_login.join
            self.receive_thread_after_login =
            (()=threading.Thread(target=self.thread_rcv_messages, args
            ()self.change_to_login_page
            : except Exception as e
            ("{"print(f"error in log out {e

        : (def clear_all_pages(self
            : try

```

```

        (self.splash_page = SplashScreen(self
        (self.sign_up_page = SignUpPage(self
        (self.forget_password_page = ForgetPasswordPage(self
            (self.login_page = LoginPage(self
            (self.main_page = MainPage(self.n, self
        (self.change_password_page = ChangePasswordPage(self
        (self.verification_code_page = VerificationCodePage(self
            :except Exception as e
            ("{print(f"error in clearing pages {e

        :(def close_all_pages(self
            :try
                ()self.main_page.close
            ()self.change_password_page.close
            ()self.verification_code_page.close
                ()self.login_page.close
            ()self.forget_password_page.close
                ()self.sign_up_page.close
                ()self.splash_page.close
            :except Exception as e
            ("{print(f"error in closing pages {e

        :(def hide_all_pages(self
            :try
                ()self.splash_page.hide
                ()self.sign_up_page.hide
            ()self.forget_password_page.hide
                ()self.login_page.hide

```



```

        ()self.main_page.hide
    ()self.change_password_page.hide
    ()self.verification_code_page.hide
    except Exception as e
    ("{print(f"error in hiding pages {e

    :(def change_to_login_page(self
    ("self.change_page("login_page

    :(def change_to_sign_up_page(self
    ("self.change_page("sign_up_page

    :(def change_to_change_password_page(self
    ("self.change_page("change_password_page

    :(def change_to_verification_code_page(self
    ("self.change_page("verification_code_page

    :(def change_to_main_page(self
    ("self.change_page("main_page

    :(def change_to_forget_password_page(self
    ("self.change_page("forget_password_page

    :(def change_to_splash_page(self
    ("self.change_page("splash_page

    :(def change_page(self, page_name

```

```

        :if self.current_page is not None
            ()self.current_page.hide
        :if page_name == "main_page
            ()self.main_page.showMaximized
            self.current_page = self.main_page
        :elif page_name == "sign_up_page
            ()self.sign_up_page.showMaximized
            self.current_page = self.sign_up_page
        :elif page_name == "change_password_page
            ()self.change_password_page.showMaximized
            self.current_page = self.change_password_page
        :elif page_name == "verification_code_page
            ()self.verification_code_page.showMaximized
            self.current_page = self.verification_code_page
        :elif page_name == "login_page
            ()self.login_page.showMaximized
            self.current_page = self.login_page
        :elif page_name == "forget_password_page
            ()self.forget_password_page.showMaximized
            self.current_page = self.forget_password_page
        :elif page_name == "splash_page
            (self.splash_page = SplashScreen(self
            ()self.splash_page.showMaximized
            self.current_page = self.splash_page

        : (def thread_recv_messages(self
            n = self.n
            ("print("receiving thread started running

```

```

                                :while self.is_logged_in
                                ()data = n.recv_str
                                :try
                                :if data is not None
                                ("message_type = data.get("message_type
                                :if message_type == "messages_list
                                ("message_list = json.loads(data.get("messages_list
                                self.main_page.list_messages = message_list
                                self.main_page.is_new_chat_clicked = True
                                self.main_page.is_messages_need_update = True
                                QMetaObject.invokeMethod(self.main_page,
                                                                ,""updated_chat_signal
                                (Qt.QueuedConnection
                                ("print("Updated the messages list
                                :elif message_type == "settings_dict
                                ("settings_dict = data.get("settings_dict

                                (self.main_page.update_settings_from_dict_signal.emit(settings_dict
                                ("print("got settings
                                :elif message_type == "playlist_current_song_bytes
                                ("audio_bytes = data.get("audio_bytes
                                ("title = data.get("title
                                play_mp3_from_bytes(audio_bytes,
                                                                (self.main_page.playlist_media_player
                                ("print(f"got song {title} bytes from server
                                :elif message_type == "searched_song_result
                                ("info_dict = data.get("searched_song_dict
                                (self.main_page.insert_search_result_signal.emit(info_dict
                                ("print("got song search info

```

```

        : "elif message_type == "playlist_songs
        ("playlist_songs_list = pickle.loads(data.get("playlist_songs_list
        self.main_page.playlist_songs = playlist_songs_list
        QMetaObject.invokeMethod(self.main_page,
                                , ""insert_playlist_to_table_signal
        (Qt.QueuedConnection
        ("print("got playlists songs
        : "elif message_type == "message_list_addition
        message_list_addition =
        (("json.loads(data.get("message_list_addition
        self.main_page.list_messages = self.main_page.list_messages +
        message_list_addition
    )self.main_page.insert_messages_into_message_box_signal.emit
        (message_list_addition
    )self.main_page.scroll_back_to_index_before_update_signal.emit
        ((len(message_list_addition
        : "elif message_type == "new_message
        ("chat = data.get("chat_name
        :if self.main_page.selected_chat == chat
        ("message_dict = json.loads(data.get("message_dict
        (self.main_page.list_messages.insert(0, message_dict

        (self.main_page.insert_new_message_in_chat_signal.emit(message_dict
        (self.main_page.chats_list.remove(chat
        (self.main_page.chats_list.insert(0, chat
        ("{} print("got new message from current chat from

        :else
        QMetaObject.invokeMethod(self.main_page,
                                , ""new_message_play_audio_signal
        (Qt.QueuedConnection

```

```

        ("print("got new message
        :elif message_type == "requests_list
        ("requests_list = json.loads(data.get("requests_list
        self.main_page.request_list = requests_list
        QMetaObject.invokeMethod(self.main_page,
                                ,""updated_social_page_signal
        (Qt.QueuedConnection
        ("print("Updated the requests list
        :elif message_type == "vc_data
        ("compressed_vc_data = data.get("compressed_vc_data
        ("speaker = data.get("speaker
        (vc_data = zlib.decompress(compressed_vc_data
        ((self.main_page.vc_data_list.append((vc_data, speaker
        :elif message_type == "share_screen_data
        compressed_share_screen_data =
        ("data.get("compressed_share_screen_data
        ("speaker = data.get("speaker
        ("frame_shape = data.get("frame_shape
        share_screen_data =
        (zlib.decompress(compressed_share_screen_data
        decompressed_frame = np.frombuffer(share_screen_data,
        (dtype=np.uint8).reshape(frame_shape

        (self.main_page.update_stream_screen_frame(decompressed_frame
        :elif message_type == "share_camera_data
        compressed_share_camera_data =
        ("data.get("compressed_share_camera_data
        ("speaker = data.get("speaker
        ("frame_shape = data.get("frame_shape
        share_screen_data =
        (zlib.decompress(compressed_share_camera_data

```

```

decompressed_frame = np.frombuffer(share_screen_data,
                                   (dtype=np.uint8).reshape(frame_shape

(self.main_page.update_stream_screen_frame(decompressed_frame
                                           : "elif message_type == "friends_list
                                           ("json_friends_list = data.get("friends_list
                                           (friends_list = json.loads(json_friends_list
                                           self.main_page.friends_list = friends_list
                                           QMetaObject.invokeMethod(self.main_page,
                                           , ""updated_social_page_signal
                                           (Qt.QueuedConnection
                                           QMetaObject.invokeMethod(self.main_page,
                                           , ""update_chat_page_without_messages_signal
                                           (Qt.QueuedConnection
                                           ("{print(f"Got friends list: {self.main_page.friends_list
                                           ("print("Updated friends_list list
                                           : "elif message_type == "online_users_list
                                           ("online_users_list = json.loads(data.get("online_users_list
                                           self.main_page.online_users_list = online_users_list
                                           QMetaObject.invokeMethod(self.main_page,
                                           , ""updated_social_page_signal
                                           (Qt.QueuedConnection
                                           QMetaObject.invokeMethod(self.main_page,
                                           , ""update_chat_page_without_messages_signal
                                           (Qt.QueuedConnection
                                           ("{print(f"Got online users list: {online_users_list
                                           : "elif message_type == "blocked_list
                                           ("blocked_list = json.loads(data.get("blocked_list
                                           self.main_page.blocked_list = blocked_list
                                           QMetaObject.invokeMethod(self.main_page,
                                           , ""updated_social_page_signal

```

```

        (Qt.QueuedConnection
            ("print("Updated the requests list
                :elif message_type == "groups_list
            ("groups_list = json.loads(data.get("groups_list
                self.main_page.groups_list = groups_list
            QMetaObject.invokeMethod(self.main_page,
                ,""update_chat_page_without_messages_signal
            (Qt.QueuedConnection
                ,QMetaObject.invokeMethod(self.main_page
                    caching_circular_images_of_groups_signal",
                                                    (Qt.QueuedConnection
                ("print("Updated the Groups list
                    :elif message_type == "chats_list
            ("chats_list = json.loads(data.get("chats_list
                self.main_page.chats_list = chats_list
            QMetaObject.invokeMethod(self.main_page,
                ,""update_chat_page_without_messages_signal
            (Qt.QueuedConnection
                ("print("Updated the chats list
            ("{print(f"chats list is: {self.main_page.chats_list
                :elif message_type == "add_chat
                ("new_chat = data.get("chat_to_add
            (self.main_page.chats_list.insert(0, new_chat
            QMetaObject.invokeMethod(self.main_page,
                ,""update_chat_page_without_messages_signal
            (Qt.QueuedConnection
                ("print("Updated the chats list
            ("{print(f"chats list is: {self.main_page.chats_list
                :elif message_type == "new_group_dict
            ("new_group_dict = json.loads(data.get("group_dict

```

```

(self.main_page.groups_list.append(new_group_dict
    QMetaObject.invokeMethod(self.main_page,
                                ,""updated_chat_signal"
                                (Qt.QueuedConnection
                                ("print("Added new group to group_list
                                :elif message_type == "call
                                ("call_action_type = data.get("call_action_type
                                :if call_action_type == "in_call_action
                                ("action = data.get("action
                                :if action == "calling
                                ("user_that_called = data.get("user_that_called
if self.main_page.is_in_a_call or self.main_page.is_calling or
                                :self.main_page.is_getting_called
                                ("print(f"User got a call but he is busy
                                :else
                                :try
                                getting_called_by = user_that_called
                                ("{print(f"User is called by {getting_called_by
                                self.main_page.is_getting_called = True
self.main_page.getting_called_by = getting_called_by
                                ,QMetaObject.invokeMethod(self.main_page
                                ,""update_chat_page_without_messages_signal"
                                (Qt.QueuedConnection
                                ,QMetaObject.invokeMethod(self.main_page
                                getting_call_signal", "
                                (Qt.QueuedConnection
                                :except Exception as e
                                ("{print(f"error in action calling error:{e

```



```

if self.main_page.is_getting_called or self.main_page.is_calling
    :or self.main_page.is_joining_call
        :if action == "rejected
            ("print(f"call was rejected
QMetaObject.invokeMethod(self.main_page,
                        ,""stop_sound_signal
            (Qt.QueuedConnection
QMetaObject.invokeMethod(self.main_page,
                        ,""reset_call_var_signal
            (Qt.QueuedConnection
                :if action == "accepted
                    ("print(f"call was accepted
QMetaObject.invokeMethod(self.main_page,
                        ,""stop_sound_signal
                    (Qt.QueuedConnection
QMetaObject.invokeMethod(self.main_page,
                        ,""initiating_call_signal
                    (Qt.QueuedConnection
QMetaObject.invokeMethod(self.main_page,
                        ,""updated_chat_signal
                    (Qt.QueuedConnection
,QMetaObject.invokeMethod(self.main_page
    start_call_threads_signal",
                                (Qt.QueuedConnection
                                    :if action == "timeout
                                        ("print("call timeout passed
QMetaObject.invokeMethod(self.main_page,
                        ,""stop_sound_signal
                    (Qt.QueuedConnection
QMetaObject.invokeMethod(self.main_page,
                        ,""reset_call_var_signal

```

```

        (Qt.QueuedConnection
            : "if action == "ended"
            ("print("call ended
        QMetaObject.invokeMethod(self.main_page,
                                ,""close_call_threads_signal

        (Qt.QueuedConnection
        QMetaObject.invokeMethod(self.main_page,
                                ,""reset_call_var_signal

        (Qt.QueuedConnection
            : "if call_action_type == "call_dictionary
            ("action = data.get("action
            : "if action == "dict
            ("call_dictionary = data.get("dict
            ("print(f'got call dict {call_dictionary
                call_dict = call_dictionary
                if
            : ("self.main_page.is_call_dict_exists_by_id(call_dict.get("call_id
            (self.main_page.update_call_dict_by_id(call_dict
            : if self.main_page.is_watching_screen
                if self.main_page.username in
                    : ("call_dict.get("participants
        : "if self.main_page.watching_type == "ScreenStream
            if self.main_page.watching_user not in
                )call_dict.get
            : ("screen_streamers"
        ,QMetaObject.invokeMethod(self.main_page
        , "stop_watching_stream_signal"
        (Qt.QueuedConnection
            : else

```

```

        if self.main_page.watching_user not in
            call_dict.get
                ("camera_streamers"
                ,QMetaObject.invokeMethod(self.main_page
                ,"stop_watching_stream_signal"
                (Qt.QueuedConnection
                :else
                (self.main_page.call_dicts.append(call_dict
                QMetaObject.invokeMethod(self.main_page,
                ,""update_chat_page_without_messages_signal
                (Qt.QueuedConnection
                :if action == "list_call_dicts
                ("list_call_dicts = json.loads(data.get("list_call_dicts
                list_of_call_dicts = list_call_dicts
                self.main_page.call_dicts = list_of_call_dicts
                ("{print(f'got list of call dicts: {list_of_call_dicts
                :if call_action_type == "update_calls
                ("action = data.get("action
                :if action == "remove_id
                ("id_to_remove = data.get("removed_id
                (self.main_page.remove_call_dict_by_id(id_to_remove
                :elif message_type == "profile_dicts_list
                ("profile_dicts_list = (data.get("profile_dicts_list
                (print(profile_dicts_list
                self.main_page.list_user_profile_dicts = profile_dicts_list
                QMetaObject.invokeMethod(self.main_page,
                ,""updated_settings_signal
                (Qt.QueuedConnection
                QMetaObject.invokeMethod(self.main_page,
                ,""update_chat_page_without_messages_signal

```

```

        (Qt.QueuedConnection
        ,QMetaObject.invokeMethod(self.main_page
        caching_circular_images_of_users_signal", "
        (Qt.QueuedConnection

        ("print("got list of profile dictionaries
        : "elif message_type == "updated_profile_dict
        (("profile_dict = json.loads(data.get("profile_dict
        ("name_of_profile_dict = data.get("username

        ,self.main_page.updating_profile_dict_signal.emit(name_of_profile_dict
        (profile_dict

        ("{"print(f"got updated profile dictionary of {name_of_profile_dict
        : "elif message_type == "update_group_dict
        (("group_dict = json.loads(data.get("group_dict
        (self.main_page.update_group_lists_by_group.emit(group_dict
        ("{'print(f"got updated group dic {group_dict.get('group_id
        : "elif message_type == "data
        ("action = data.get("action
        : "if action == "receive
        ("receive_status = data.get("receive_status
        : "if receive_status == "done
        QMetaObject.invokeMethod(self.splash_page,
        ,""stop_loading_signal

        (Qt.QueuedConnection

        : "elif message_type == "security_token
        ("security_token = data.get("security_token
        (save_token(security_token

        : "elif message_type == "friend_request
        ("status = data.get("friend_request_status

```

```

        : "if status == "not exist
    ()self.main_page.friends_box.friend_not_found
        : "elif status == "already friends
    ()self.main_page.friends_box.request_was_friend
        : "elif status == "worked
    ()self.main_page.friends_box.request_was_sent
        : "elif status == "active
    ()self.main_page.friends_box.request_is_pending
        : except Exception as e
    (" {print(f"error in receiving thread {e
    (" print("thread receive messages ended

        : ' __if __name__ == '__main
    ()page_controller_object = PageController

import socket
import threading
from server_net import ServerNet
* from email_send_code import
from server_handling_classes import ServerHandler
import database_func
import re
import random
import json
import time
import zlib
import logging

```

```

import base64

from song_search_engine import extract_audio_bytes

import datetime


Set up the logging configuration #
, "logging.basicConfig(filename="example.log
, level=logging.INFO
, 'format'%(asctime)s - %(levelname)s - %(message)s
('datefmt='%Y-%m-%d %H:%M:%S
()console_handler = logging.StreamHandler
(console_handler.setLevel(logging.INFO
formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s',
('"%Y-%m-%d %H:%M:%S
(console_handler.setFormatter(formatter


Add the StreamHandler to the root logger #
(logging.getLogger().addHandler(console_handler


"server = "0.0.0.0
port = 5555
(tcp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM
(udp_server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM
((udp_server_socket.bind((server, port


ringing_list made by tuples of len = 2 the [0] in the tuple is the caller and the [1] #
is the one being called

[] = ringing_list

messages_to_clients_list made by tuples of len = 2 the [0] in the tuple is the #
receiver of the message

```

```

        is the message action [1] #
        {} = messages_to_clients_dict
current_calls_list made by tuples of len = 2 that represent 2 users that are in #
        .call
        [] = current_calls_list
        [] = online_users

```

```

made of tuples where [0] is target and [1] is the vc_Data #
        [] = list_vc_data_sending

```

```

:(def is_json_serialized(file_path
        :try
:with open(file_path, 'r') as file
        (json.load(file
            return True
:except json.JSONDecodeError
        return False
:except FileNotFoundError
        return False

```

```

:(def create_song_info_dict(audio_bytes, title, thumbnail_bytes, audio_duration
        } = video_info
        ,audio_bytes": audio_bytes"
        ,title": title"
        ,thumbnail_bytes": thumbnail_bytes"
        audio_duration": audio_duration"
        {

```

```
return video_info
```

```
:(def is_string(variable  
  (return instanceof(variable, str
```

```
:(def is_dict(variable  
  (return instanceof(variable, dict
```

```
:(def is_zlib_compressed(data  
  :try
```

```
    Attempt to decompress the data using zlib #
```

```
    (zlib.decompress(data
```

```
      return True
```

```
    :except zlib.error
```

```
    If an error occurs during decompression, it's not zlib compressed #
```

```
    return False
```

```
:(def is_client_waits_for_message(client  
  global messages_to_clients_dict
```

```
    Use the 'in' operator to check if the client is in the dictionary keys #
```

```
    return client in messages_to_clients_dict
```

```
:(def add_message_for_client(client, message
```



```

        global messages_to_clients_dict
        :if client in ServerHandler.online_users
        messages_to_clients_dict[client] = message

:(def get_and_remove_message_for_client(client
    global messages_to_clients_dict
    Use the pop method to retrieve and remove the message for the client #
    If the client is not found, return None #
    (return messages_to_clients_dict.pop(client, None

:(def gets_group_attributes_from_format(group_format
    ("")parts = group_format.split
    [id = parts[0][1
    [name = parts[1
    return name, id

:(def parse_group_caller_format(input_format
    Define a regular expression pattern to capture the information #
    ('(\(+(())\))\(+([()^\])\(+pattern = re.compile(r'\(\\d

    Use the pattern to match the input_format #
    (match = pattern.match(input_format

    :if match
    Extract the matched groups #

```

```

        ((group_id = int(match.group(1
        )group_name = match.group(2).strip
        )group_caller = match.group(3).strip

    return group_id, group_name, group_caller
        :else
        Return None if no match is found #
        return None

:(def threaded_logged_in_client(n, User
    global list_vc_data_sending
    "" = client_current_chat
    (__logger = logging.getLogger(__name
    numbers_of_starter_message = 20
    messages_list_max_index = numbers_of_starter_message
    :(while ServerHandler.is_user_online(User
        (time.sleep(0.05
    :(if is_client_waits_for_message(User
    (message = get_and_remove_message_for_client(User
        :(if isinstance(message, dict
            ("message_type = message.get("message_type
            :("if message_type == "current_chat
            ("client_current_chat = message.get("current_chat
            messages_list_max_index = 20
            ("__logger.info(f"{User} current chat is {client_current_chat
    :(if client_current_chat not in database_func.get_user_chats(User
        (database_func.add_chat_to_user(User, client_current_chat

```

```

        (n.add_new_chat(client_current_chat
        ("{logger.info(f"added new chat to {User
                                list_dict_of_messages =
database_func.get_last_amount_of_messages(User, client_current_chat, 0
        ,
                                (messages_list_max_index
(n.send_messages_list(list_dict_of_messages
        : "elif message_type == "add_message
        ("sender = message.get("sender
        : if sender == client_current_chat
        ("content = message.get("content
        ("type_of_message = message.get("type
        ("file_name = message.get("file_name
        ()time_now = datetime.datetime.now
('formatted_time = time_now.strftime('%Y-%m-%d %H:%M
        } = formatted_message
        ,content": content"
        ,sender_id": sender"
        ,timestamp": formatted_time"
        ,message_type": type_of_message"
        file_name": file_name"
                                {
(n.send_new_message_content(sender, formatted_message
        ("{logger.info(f"send new message to {User
        : "if message_type == "more_messages
                                list_dict_of_messages =
        ,database_func.get_last_amount_of_messages(User, client_current_chat
(messages_list_max_index + 1, messages_list_max_index + 6
                                messages_list_max_index += 6

```

```

        :if len(list_dict_of_messages) > 0
(n.send_addition_messages_list(list_dict_of_messages
    ("{logger.info(f'sent more messages to {User
        :(if isinstance(message, list
            ("{logger.info(f'Sent online users list to {User
        (friends_list = database_func.get_user_friends(User
            online_list = message
        ((message = list(set(friends_list) & set(online_list
            (n.send_online_users_list(message
                :if message == "friends_request:send
(friend_request_list = database_func.get_friend_requests(User
            (n.send_requests_list(friend_request_list
        ("{print(f'Sent requests list ({friend_request_list}) to user {User
            :if message == "friends_list:send
        (friends_list = database_func.get_user_friends(User
            (n.send_friends_list(friends_list
        ("{logger.info(f'Sent friend list ({friends_list}) to user {User

:(def thread_recv_messages(n, addr
    "" = User
    is_logged_in = False
    (__logger = logging.getLogger(__name
        :while True
            :if not is_logged_in
                :try
                    ()data = n.recv_str
                :if data is None

```

```

break
("message_type = data.get("message_type
    :if message_type == "login
        ("username = data.get("username
            ("password = data.get("password
                (is_valid = database_func.login(username, password
                    :if is_valid
                        (user_settings = database_func.get_user_settings(username
                            :if user_settings
                                ("is_2fa_for_user = user_settings.get("two_factor_auth
                                    :else
                                        is_2fa_for_user = False
                                    :if not username in ServerHandler.online_users
                                        :if not is_2fa_for_user
                                            ()n.send_confirm_login
                                        ("{logger.info(f"Server sent Confirmed to client {username
                                            ("logger.info(f"Client {username} logged in
                                                User = username
                                                (ServerHandler.user_online(User, n
                                                    is_logged_in = True
threading.Thread(target=threaded_logged_in_client, args=(n,
                                                                    ()User)).start
                                                                    :else
                                                                    user_mail =
                                                                    (database_func.get_email_by_username(username
                                                                    ("logger.info(f"{username} has 2fa On
                                                                    (code = random.randint(100000, 999999
                                                                    send_login_code_to_client_email(code, user_mail,
                                                                    (username

```

```

                                ()n.send_2fa_on
attempts_remaining = 3 # Set the maximum number of
                                attempts

                                :while attempts_remaining > 0
                                ()data = n.recv_str
                                ("message_type = data.get("message_type
                                :if message_type == "login
                                ("action = data.get("action
                                :if action == "2fa
                                ("code_gotten = data.get("code
                                :if int(code_gotten) == code
                                ()n.send_2fa_code_valid
                                ("logger.info(f"got right 2fa code from {username
                                logger.info(f"Server sent Confirmed to client
                                ("{{username

                                ("logger.info(f"Client {username} logged in
                                User = username
                                (ServerHandler.user_online(User, n
                                is_logged_in = True

                                ,threading.Thread(target=threaded_logged_in_client
                                ()args=(n, User)).start

                                break
                                :else
                                attempts_remaining -= 1

                                :else
                                ()n.send_already_logged_in
                                logger.info(f"{username} already logged in from another device,
                                ("cannot log in from 2 devices
                                :else

```

```

({logger.info(f"Server sent Invalid to address ({addr
    (n.send_invalid_login
    :if message_type == "sign_up
    ("username = data.get("username
    ("password = data.get("password
    ("email = data.get("email
    (is_valid = not database_func.username_exists(username
    :if is_valid

    ({logger.info(f"Server sent code to email for ({addr
        (code = random.randint(100000, 999999
    (send_sign_up_code_to_client_email(code, email, username
        (n.send_code_to_mail

attempts_remaining = 3 # Set the maximum number of attempts
    :while attempts_remaining > 0
        ()data = n.recv_str
        ("message_type = data.get("message_type
        :if message_type == "sign_up
            ("action = data.get("action
            :if action == "verification_code
                ("code_gotten = data.get("code
                :if code_gotten is None
                    ({logger.info(f"lost connection with ({addr
                        break

                    (code_gotten = int(code_gotten

    ({logger.info(f"Server got {code_gotten} from ({addr
        :if code_gotten == code
    (send_confirmation_to_client_email(email, username
    ({logger.info(f"Server sent Confirmed to ({addr

```

```

        ()n.send_sign_up_code_valid
(database_func.insert_user(username, password, email
        ("logger.info(f"inserted: {username} to data base
                                break
        :elif code_gotten == "sign_up:cancel
        ("logger.info(f"({addr}) existed code menu
                                break
                                :else
        ("({logger.info(f"Server sent sign_up Invalid to ({addr
        ()n.send_sign_up_code_invalid
        attempts_remaining -= 1
                                :else
        ("({logger.info(f"Server sent Invalid to ({addr
        ()n.send_sign_up_invalid
        :if message_type == "forget password
        ("username = data.get("username
        ("email = data.get("email
        (is_valid = database_func.user_exists_with_email(username, email
                                :if is_valid
        ("({logger.info(f"Server sent code to email for ({addr
        (code = random.randint(100000, 999999
        (send_forget_password_code_to_email(code, email, username
        ()n.send_forget_password_info_valid
attempts_remaining = 3 # Set the maximum number of attempts
                                :while attempts_remaining > 0
        ()code_gotten_data = n.recv_str
        ("message_type = code_gotten_data.get("message_type
        :if message_type == "sign_up

```



```

        ("action = code_gotten_data.get("action
            :if action == "verification_code
("code_gotten = code_gotten_data.get("code
("({logger.info(f"got {code_gotten} from ({addr
            :if code_gotten is None
("({logger.info(f"lost connection with ({addr
                break
            (code_gotten = int(code_gotten
("({logger.info(f"Server got {code_gotten} from ({addr
            :if code_gotten == code
("logger.info(f"code gotten from {username} is correct
        (n.send_forget_password_code_valid
            ()data = n.recv_str
("message_type = data.get("message_type
        :if message_type == "password
            ("action = data.get("action
            :if action == "new_password
("new_password = data.get("new_password
database_func.change_password(username,
                                (new_password
                                send_changed_password_to_email(email,
                                (username
("logger.info(f"{username} changed password
                break
            :elif code_gotten == "Exit
("logger.info(f"({addr}) existed code menu
                break
            :else

```

```

        logger.info(f"Server sent Invalid to ({addr}) because
                                                                ("code was incorrect
                                                                (n.send_forget_password_code_invalid
                                                                attempts_remaining -= 1
                                                                :else
("logger.info("Server sent Invalid (Username with email don't exist
                                                                (n.send_forget_password_info_invalid
                                                                :if message_type == "security_token
                                                                ("security_token = data.get("security_token
(username = database_func.check_security_token(security_token
                                                                :if not username
("logger.info(f"security token from ({addr}) isn't valid
                                                                (n.send_security_token_invalid
                                                                :else
                                                                :if username not in online_users
                                                                (n.send_security_token_valid
                                                                User = username
                                                                (n.send_username_to_client(username
                                                                ("logger.info(f"{User} logged in
                                                                (ServerHandler.user_online(User, n
                                                                is_logged_in = True
threading.Thread(target=threaded_logged_in_client, args=(n,
                                                                (User))).start
                                                                :else
                                                                (n.send_username_to_client_login_invalid(username
logger.info(f"{username} already logged in from another device,
                                                                ("cannot log in from 2 devices
("logger.info(f"Blocked User from logging in from 2 devices

```

```

                                :except Exception as e
                                    (print(e
                                :else
                                    ("{logger.debug(f"waiting for data...for {User #
                                    ()data = n.recv_str
                                    :if data is None
                                    ("{logger.info(f"lost connection with {User
                                        (ServerHandler.user_offline(User
                                            break
                                    ("message_type = data.get("message_type
                                        :if message_type == "connect_udp_port
                                    ("udp_address = data.get("udp_address
                                    ("tcp_address = data.get("tcp_address
                                        :if ServerHandler.server_mtu is None
                                    (ServerHandler.check_max_packet_size_udp(udp_address
                                        ServerHandler.create_and_add_udp_handler_object(User,
                                            (udp_address, tcp_address
                                        :elif message_type == "logout
                                            ("{logger.info(f"logged out {User
                                            (ServerHandler.user_offline(User
                                                is_logged_in = False
                                                "" = User
                                            (n.timeout_receive
                                        :elif message_type == "playlist_song_bytes_by_index
                                            ("index = data.get("index
song_dict = database_func.get_song_by_index_and_owner(User,
                                                                    (index
                                            ('audio_bytes = song_dict.get('audio_bytes
                                            ('title = song_dict.get('title

```

```

        (n.send_played_song_bytes(audio_bytes, title
                                : "elif message_type == "exit_group"
                                ("group_to_exit_id = data.get("group_to_exit_id"
(database_func.remove_group_member(group_to_exit_id, User
                                group_name =
                                (database_func.get_group_name_by_id(group_to_exit_id
                                "{group_name_plus_id = f"({group_to_exit_id}){group_name
(database_func.remove_chat_from_user(User, group_name_plus_id
                                : "elif message_type == "remove_user_from_group"
                                ("user_to_remove = data.get("user_to_remove"
                                ("group_id = data.get("group_id"
(database_func.remove_group_member(group_id, user_to_remove
                                (group_name = database_func.get_group_name_by_id(group_id
                                "{group_name_plus_id = f"({group_id}){group_name
                                database_func.remove_chat_from_user(user_to_remove,
                                (group_name_plus_id
                                : "elif message_type == "remove_chat"
                                ("chat_to_remove = data.get("chat_to_remove"
(database_func.remove_chat_from_user(User, chat_to_remove
                                ("{logger.info(f"remove chat {chat_to_remove} for {User
                                : "elif message_type == "settings_dict"
                                ("settings_dict = data.get("settings_dict"
(database_func.update_settings_by_dict(User, settings_dict
                                ("logger.info(f"updated {User} settings"
                                : "elif message_type == "current_chat"
                                ("user_current_chat = data.get("current_chat"
                                (add_message_for_client(User, data
                                : "elif message_type == "song_search"
                                ("search_str = data.get("search_str"

```

```

        ("{logger.info(f'searching for {search_str} for {User
                                                                    :try
audio_bytes, video_title, thumbnail_bytes, duration_min_sec =
                                                                    (extract_audio_bytes(search_str
info_dict = create_song_info_dict(audio_bytes, video_title,
                                                                    (thumbnail_bytes, duration_min_sec
                                                                    (n.send_searched_song_info(info_dict
                                                                    :except Exception as e
        ("{logger.error(f'error with search engine: {e
                                                                    :elif message_type == "remove_song
                                                                    ("song_title = data.get("song_title
(database_func.remove_song(song_title, User
                                                                    :elif message_type == "save_song
                                                                    ("song_dict = data.get("song_dict
                                                                    ("audio_bytes = song_dict.get("audio_bytes
                                                                    ("title = song_dict.get("title
                                                                    ("thumbnail_bytes = song_dict.get("thumbnail_bytes
                                                                    ("audio_duration = song_dict.get("audio_duration
database_func.add_song(title, audio_bytes, User, audio_duration,
                                                                    (thumbnail_bytes
        ("logger.info(f'Added song {title} to {User} playlist
                                                                    :elif message_type == "more_messages
                                                                    (add_message_for_client(User, data
                                                                    :elif message_type == "call
                                                                    ("call_action_type = data.get("call_action_type
                                                                    :if call_action_type == "stream
                                                                    ("stream_type = data.get("stream_type
                                                                    ("stream_action = data.get("action
                                                                    :if stream_action == "start

```

```

ServerHandler.create_video_stream_for_user_call(User,
                                                (stream_type
:"if stream_action == "close
ServerHandler.close_video_stream_for_user_call(User,
                                                (stream_type
:"if stream_action == "watch
("streamer = data.get("user_to_watch
ServerHandler.add_spectator_to_call_stream(User, streamer,
                                                (stream_type
:"if stream_action == "stop_watch
(ServerHandler.remove_spectator_from_call_stream(User
:"if call_action_type == "in_call_action
("action = data.get("action
:"if action == "join
("group_id = data.get("group_id
:(if ServerHandler.is_user_in_a_call(User
(ServerHandler.remove_user_from_call(User
(ServerHandler.add_user_to_group_call_by_id(User, group_id
:"else
(ServerHandler.add_user_to_group_call_by_id(User, group_id
:"if action == "mute_myself
(ServerHandler.mute_or_unmute_self_user(User
:"if action == "deafen_myself
(ServerHandler.deafen_or_undeafen_self_user(User
:"if action == "calling
("user_that_is_getting_called = data.get("calling_to
("{logger.info(f"{User} calling {user_that_is_getting_called
(ServerHandler.create_ring(User, user_that_is_getting_called
:"if action == "accepted_call

```

```

        ("ringer = data.get("accepted_caller
        ("logger.info(f"{User} accepted {ringer} call
if a call already created no need to create just need to append #
                                the user to the call

                                :(")")if ringer.startswith
                                group_id, group_name, caller =
                                    (parse_group_caller_format(ringer
                                :("if ServerHandler.is_group_call_exist_by_id(group_id
                                ServerHandler.add_user_to_group_call_by_id(User,
                                                                    (group_id
                                :else
([ServerHandler.create_call_and_add(group_id, [User, caller
                                :else
([ServerHandler.create_call_and_add(None, [User, ringer
    (ServerHandler.accept_ring_by_ringer(ringer, User
        :("if action == "rejected_call
        ("rejected_caller = data.get("rejected_caller
        :(")")if rejected_caller.startswith
            group_id, group_name, caller =
                (parse_group_caller_format(rejected_caller
logger.info(f"{User} rejected {caller} Group call, Group:
                                                                    ("{{group_name
        (ServerHandler.reject_ring_by_ringer(caller, User
                                :else
            rejected_caller = rejected_caller
        ("logger.info(f"{User} rejected {rejected_caller} call
        (ServerHandler.reject_ring_by_ringer(rejected_caller, User
                                :("if action == "ended
        (ServerHandler.remove_user_from_call(User
            :("if call_action_type == "change_calling_status

```

```

        ("action = data.get("action
            :!"if action == "stop
(ServerHandler.cancel_ring_by_the_ringer(User
        :elif message_type == "add_message
            ("sender = data.get("sender
            ("receiver = data.get("receiver
            ("content = data.get("content
            ("type_of_message = data.get("type
            ("file_name = data.get("file_name
        database_func.add_message(sender, receiver, content,
                                (type_of_message, file_name
                                ...fix it #
                                :(")")if not receiver.startswith
(ServerHandler.update_message_for_users([receiver], data, User
                                :else
                                group_name, group_id =
                                (gets_group_attributes_from_format(receiver
(group_members = database_func.get_group_members(group_id
                                (print(group_members
                                (group_members.remove(User
ServerHandler.update_message_for_users(group_members, data,
                                (receiver

        ("logger.info(f"added new message from {User} to {receiver
        :elif message_type == "update_profile_pic
        ("b64_encoded_profile_pic = data.get("b64_encoded_profile_pic
        :if b64_encoded_profile_pic == "None
            b64_encoded_profile_pic = None
        ("logger.info(f"{User} reset his profile picture
(database_func.update_profile_pic(User, b64_encoded_profile_pic

```



```

        ({logger.info(f"updated client profile pic of {User
ServerHandler.update_profiles_list_for_everyone_by_user(User,
                                                    (b64_encoded_profile_pic
                                                    :elif message_type == "security_token
                                                    ("action = data.get("action
                                                    :if action == "needed
(user_security_token = database_func.get_security_token(User
        (n.send_security_token_to_client(user_security_token
({logger.info(f"Sent security token to - {User} , {user_security_token
                                                    :elif message_type == "vc_data
("compressed_vc_data = data.get("compressed_vc_data
        :if compressed_vc_data is not None
        (vc_data = zlib.decompress(compressed_vc_data
        (ServerHandler.send_vc_data_to_call(vc_data, User
        :elif message_type == "share_screen_data
        compressed_share_screen_data =
        ("data.get("compressed_share_screen_data
        ("shape_of_frame = data.get("shape_of_frame
        :if compressed_share_screen_data is not None
        share_screen_data =
        (zlib.decompress(compressed_share_screen_data
        ServerHandler.send_share_screen_data_to_call(share_screen_data,
        ("shape_of_frame, User, "ScreenStream
        :elif message_type == "share_camera_data
        compressed_share_camera_data =
        ("data.get("compressed_share_camera_data
        ("shape_of_frame = data.get("shape_of_frame
        :if compressed_share_camera_data is not None
        share_screen_data =
        (zlib.decompress(compressed_share_camera_data

```

```

ServerHandler.send_share_screen_data_to_call(share_screen_data,
                                             ("shape_of_frame, User, "CameraStream
                                             :elif message_type == "friend_request
("username_for_request = data.get("username_for_request
                                   user = User
                                   friend_user = username_for_request
if not database_func.are_friends(user, friend_user) and
                                   )database_func.username_exists
friend_user) and not friend_user == user and not
                                   ,database_func.is_active_request(user
:(friend_user
    (database_func.send_friend_request(user, friend_user
    ("logger.info(f"{user} sent friend request to {friend_user
("add_message_for_client(friend_user, "friends_request:send
    ("n.sent_friend_request_status("worked
                                   :else
:(if database_func.is_active_request(user, friend_user
    ("logger.info("friend request is active
    ("n.sent_friend_request_status("active
:(elif not database_func.username_exists(friend_user
    ("n.sent_friend_request_status("not exist
                                   :else
    ("n.sent_friend_request_status("already friends
    :elif message_type == "friend_request_status
    ("status = data.get("action
    :if status == "accept
("accepted_or_rejected_user = data.get("accepted_user
    database_func.handle_friend_request(User,
    (accepted_or_rejected_user, True

```

```

logger.info(f"{User} accepted {accepted_or_rejected_user} friend
                                                    ("request

("add_message_for_client(User, "friends_request:send
add_message_for_client(accepted_or_rejected_user,
                        ("friends_request:send

("add_message_for_client(User, "friends_list:send
add_message_for_client(accepted_or_rejected_user,
                        ("friends_list:send

:else

("accepted_or_rejected_user = data.get("rejected_user
    database_func.handle_friend_request(User,
        (accepted_or_rejected_user, False

logger.info(f"{User} rejected {accepted_or_rejected_user} friend
                                                    ("request

("add_message_for_client(User, "friends_request:send
add_message_for_client(accepted_or_rejected_user,
                        ("friends_request:send

:elif message_type == "friend_remove

("friends_to_remove = data.get("username_to_remove
(database_func.remove_friend(User, friends_to_remove
    :if friends_to_remove in online_users

("add_message_for_client(friends_to_remove, "friends_list:send

("logger.info(f"{User} removed {friends_to_remove} as friend

:elif message_type == "block

("user_to_block = data.get("user_to_block
(database_func.block_user(User, user_to_block
    ("{logger.info(f"{User} blocked {user_to_block

:elif message_type == "unblock

("user_to_unblock = data.get("user_to_unblock
(database_func.unblock_user(User, user_to_unblock

```

```

        ("logger.info(f'{User} unblocked {user_to_unblock
            :elif message_type == "group
                ("action = data.get("action
                    :if action == "create
                        ("members_list = json.loads(data.get("group_members_list
                            (members_list.append(User
                                group_chat_name, new_group_id =
                                    (database_func.create_group(f'{User}'s Group", User, members_list
                                        (ServerHandler.send_new_group_to_members(new_group_id
                                            (n.add_new_chat(group_chat_name
                                                ("logger.info(f'{User} created a new group
                                                    :elif action == "update_image
                                                        ("group_id = data.get("group_id
                                                            ("encoded_b64_image = data.get("encoded_b64_image
                                                                (image_bytes = base64.b64decode(encoded_b64_image
                                                                    (database_func.update_group_image(int(group_id), image_bytes
                                                                        (ServerHandler.update_group_dict_for_members(group_id
                                                                            logger.info(f'Update group image of id: {group_id} was updated by
                                                                                                    ("{{User
                                                                                                        :elif action == "add_user
                                                                                                            ("group_id = data.get("group_id
                                                                                                                ("users_to_add = json.loads(data.get("users_to_add
                                                                                                                    (group_dict = database_func.get_group_by_id(group_id
                                                                                                                        ("group_manager = group_dict.get("group_manager
                                                                                                                            :if group_manager == User
                                                                                                                                :for user in users_to_add
                                                                                                                                    (database_func.append_group_member(group_id, user
                                                                                                                                        logger.info(f'Added {users_to_add} to group of id {group_id} by
                                                                                                                                                                    ("{{User

```

```

:else
logger.critical(f'{User} tried to add user to group where he has no
("permissions

```

```

()ServerHandler = ServerHandler

```

```

:()def tcp_server
(__logger = logging.getLogger(__name
:try
((tcp_server_socket.bind((server, port
:except socket.error as e
(logger.critical(e
(tcp_server_socket.listen(20
(logger.info("Waiting for a connection , Server started
:while True
()conn, addr = tcp_server_socket.accept
("{logger.info(f'connect to: {addr
(n = ServerNet(conn, addr
()threading.Thread(target=thread_recv_messages, args=(n, addr)).start

```

```

:(def handle_udp_message(data, address
("{(print(f'got message from {address} of size {len(data

```

```

:()def listen_udp
    ServerHandler.udp_socket = udp_server_socket
    :while True
        :try
(fragment_data, address = udp_server_socket.recvfrom(100000
    (ServerHandler.handle_udp_fragment(fragment_data, address
        :except OSError as os_err
            ("print(f"OS error: {os_err
                :except Exception as e
                    ("print(f"Exception: {e

:()def main
    ()database_func.create_tables_if_not_exist #
    (tcp_thread = threading.Thread(target=tcp_server
        ()tcp_thread.start
    (udp_thread = threading.Thread(target=listen_udp
        ()udp_thread.start

: ' __if __name__ == ' __main
    ()main

* from PyQt5.QtWidgets import
from PyQt5.QtGui import QColor
    from functools import partial
from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit, QGraphicsBlurEffect
    from PyQt5.QtCore import QSize, Qt, QUrl
from PyQt5.QtGui import QIcon, QPixmap, QImage, QPainter, QPainterPath,
    QFont

```

```

from PyQt5.QtMultimedia import QMediaContent
        from io import BytesIO
            import base64
            import zlib
        from PIL import Image
            import tempfile
            import os
            import math
        import subprocess
            import random
            import string
        import pyaudio
            import cv2

        :(def is_base64_encoded(s
            :try
                Attempt to decode the Base64 string #
                (decoded_bytes = base64.b64decode(s
                    return True
                :except
                If decoding fails, it's not a valid Base64 string #
                    return False

        :(def try_to_open_output_stream(index
            audio_format = pyaudio.paInt16
                channels = 1

```

```

        rate = 44100
        chunk = 1024
        ()p = pyaudio.PyAudio
        :try
output_stream = p.open(format=audio_format, channels=channels,
                        ,rate=rate, output=True, frames_per_buffer=chunk
                        (output_device_index=index
                        return True
                        :except
        return False

```

```

:(def try_to_open_input_stream(index
    audio_format = pyaudio.paInt16
        channels = 1
        rate = 44100
        chunk = 1024
        ()p = pyaudio.PyAudio
        :try
,input_stream = p.open(format=audio_format
                        ,channels=channels
                        ,rate=rate
                        ,input=True
                        ,frames_per_buffer=chunk
                        (input_device_index=index
                        return True
                        :except
        return False

```



```

:(def replace_non_space_with_star(string1
    " = result
    :for char in string1
    :()if char != ' ' and not char.isspace
        '*' =+ result
    :else
        result += char
    return result

```

```

:(def generate_random_filename(extension
    Generate a random string of characters #
    random_string = ".join(random.choices(string.ascii_lowercase + string.digits,
                                          ((k=8
    "{return f"file_{random_string}.{extension

```

```

:(def open_pptx_from_bytes(pptx_bytes
    ('temp_file_path = save_bytes_to_temp_file(pptx_bytes, 'pptx
    (open_file_with_default_app(temp_file_path

```

```

:(def open_docx_from_bytes(docx_bytes
    ('temp_file_path = save_bytes_to_temp_file(docx_bytes, 'docx
    (open_file_with_default_app(temp_file_path

```

```

:(def open_xlsx_from_bytes(xlsx_bytes

```

```

('temp_file_path = save_bytes_to_temp_file(xlsx_bytes, 'xlsx
    (open_file_with_default_app(temp_file_path

:(def open_py_from_bytes(py_bytes
('temp_file_path = save_bytes_to_temp_file(py_bytes, 'py
    (open_file_with_default_app(temp_file_path

:(def open_pdf_from_bytes(pdf_bytes
('temp_file_path = save_bytes_to_temp_file(pdf_bytes, 'pdf
    (open_file_with_default_app(temp_file_path

:(def save_bytes_to_temp_file(file_bytes, extension
(temp_file = tempfile.NamedTemporaryFile(suffix='.' + extension, delete=False
    (temp_file.write(file_bytes
    ()temp_file.close
    return temp_file.name

:(def open_file_with_default_app(file_path
    :try
        :if os.name == 'nt
            (os.startfile(file_path
        :elif os.name == 'posix
            ([subprocess.run(['xdg-open', file_path
    :else

```

```

        (").print("Unsupported operating system
                    :except Exception as e
        ("){print(f"Error opening file: {e

:(def open_text_file_from_bytes(file_bytes
                                :try
                                Create a temporary file #
:with tempfile.NamedTemporaryFile(delete=False, suffix='.txt') as temp_file
    Write the bytes to the temporary file #
        (temp_file.write(file_bytes
                        ()temp_file.flush

        Get the path to the temporary file #
        file_path = temp_file.name

    Open the temporary file using Notepad asynchronously #
    ([subprocess.Popen(['notepad', file_path

([On macOS, you might use: subprocess.Popen(['open', file_path #
                                :except Exception as e
        ("){print(f"Error opening text file: {e

:(def download_file_from_bytes(file_bytes, file_extension, file_name
                                :try
                                Get the path to the user's downloads directory #
('downloads_dir = os.path.join(os.path.expanduser('~'), 'Downloads

```

```

                                                    :if not file_name
(file_name = generate_random_filename(file_extension

(path = os.path.join(downloads_dir, file_name

                                                    :if path
Write the file bytes to disk #
:with open(path, 'wb') as file
(file.write(file_bytes

('{print(f"File downloaded successfully to '{path
                                                    :else
('{print("Download canceled by user
:except Exception as e
('{print(f"Error downloading file: {e

:(def play_mp3_from_bytes(mp3_bytes, media_player
                                                    :try
Save MP3 bytes to a temporary file #
:if mp3_bytes is not None
(media_player.stop
('temp_file_path = save_bytes_to_temp_file(mp3_bytes, 'mp3

```

Create QMediaContent object with the URL pointing to the temporary # file

```
((media_content = QMediaContent(QUrl.fromLocalFile(temp_file_path
```

Create QMediaPlayer instance and set the media content #

```
(media_player.setMedia(media_content
```

Play the media #

```
()media_player.play
```

```
:else
```

```
("print("got None object instead of bytes can't play
```

```
:except Exception as e
```

```
("{print(f"Error playing MP3: {e
```

```
:(def format_label_text_by_row(label, text, num_rows
```

```
:try
```

Calculate the total number of characters per row, including the possibility #
of a shorter last row

```
chars_per_row = len(text) // num_rows
```

```
extra_chars = len(text) % num_rows
```

Initialize variables for storing formatted text and the starting index #

```
"" = formatted_text
```

```
start_index = 0
```

Iterate through each row #

```
:(for row in range(num_rows
```

Calculate the end index for this row #

```
end_index = start_index + chars_per_row
```

```
Add an extra character to this row if needed #
```

```
:if row < extra_chars
```

```
end_index += 1
```

```
Add the substring to the formatted text with a newline character #
```

```
"formatted_text += text[start_index:end_index] + "\n"
```

```
Update the starting index for the next row #
```

```
start_index = end_index
```

```
Set the formatted text to the label #
```

```
(label.setText(formatted_text
```

```
:except Exception as e
```

```
("{print(f" error in creating formatted label by rows: {e
```

```
:(def make_q_object_clear(object
```

```
(";object.setStyleSheet("background-color: transparent; border: none
```

```
:(def extract_first_frame(video_bytes
```

```
:try
```

```
Write video bytes to a temporary file #
```

```
(temp_video_path = tempfile.NamedTemporaryFile(delete=False
```

```
(temp_video_path.write(video_bytes
```

```
()temp_video_path.close
```

```

        Open the temporary video file #
        (cap = cv2.VideoCapture(temp_video_path.name

        Read the first frame #
        ()ret, frame = cap.read

Release the video capture object and delete the temporary file #
        ()cap.release
        ()cv2.destroyAllWindows
        ()temp_video_path.close

        Convert the first frame to bytes #
        (retval, buffer = cv2.imencode('.png', frame
        :if retval
        ()return buffer.tobytes
        :else
        return None

        :except Exception as e
        ("{print(f"Error extracting first frame: {e
        return None

:(def open_image_bytes(image_bytes
        :try
('temp_file_path = save_bytes_to_temp_file(image_bytes, 'png
        (open_file_with_default_app(temp_file_path

```

```

        :except Exception as e
        ("{print(f"Error opening image: {e
        return False

:(def create_custom_circular_label(width, height, parent
    (label = QLabel(parent

    (label_size = QSize(width, height
    (label.setFixedSize(label_size

        """))label.setStyleSheet
        } QLabel
border-radius: "" + str(height // 2) + ""px; /* Set to half of the label height
/*
background-color: transparent; /* Make the background color transparent
/*

{
    (""

    return label

:(def is_valid_image(image_bytes
    :try

    Use Pillow to try opening the image from bytes #
    ((image = Image.open(BytesIO(image_bytes
    If successful, it's a valid image #

```



```

        return True
    except Exception as e
        If there is an exception, it's not a valid image #
        ("{print(f"Error: {e
        return False

:(def file_to_bytes(file_path
:with open(file_path, "rb") as file
    ()image_bytes = file.read
    return image_bytes

:()def check_active_cameras
    :try
        (cap = cv2.VideoCapture(0
        :()if cap.isOpened
            ()cap.release
            return True
        :else
            return False
    except Exception as e
        ("{print(f"could not find active camera, error: {e
        return False

works very well for a circular labels #
:(def set_icon_from_bytes_to_label(label, image_bytes

```

```

Load the image from bytes #
                                :try
                                ()pixmap = QPixmap
(pixmap.loadFromData(image_bytes

Get the size of the label #
                                ()label_size = label.size
                                ()label_width = label_size.width
                                ()label_height = label_size.height

Calculate the aspect ratio of the image #
                                ()image_width = pixmap.width
                                ()image_height = pixmap.height
image_aspect_ratio = image_width / image_height

Determine how to scale the image based on its aspect ratio #
                                :if image_aspect_ratio <= 0.5
scaled_pixmap = pixmap.scaledToWidth(label_width,
                                    (Qt.SmoothTransformation
                                :elif image_aspect_ratio >= 1.5
scaled_pixmap = pixmap.scaledToHeight(label_height,
                                    (Qt.SmoothTransformation
                                :else
scaled_pixmap = pixmap.scaled(label_size, Qt.KeepAspectRatio,
                                (Qt.SmoothTransformation

Set the scaled pixmap to the label #
                                (label.setPixmap(scaled_pixmap
                                (label.setAlignment(Qt.AlignCenter

```

```

        :except Exception as e
        ("{print(f"error in loading image from bytes {e

:(def set_icon_from_path_to_label(label, image_path
    Load the image from file path #
    (pixmap = QPixmap(image_path

    Get the size of the label #
    ()label_size = label.size
    ()label_width = label_size.width
    ()label_height = label_size.height

    Calculate the aspect ratio of the image #
    ()image_width = pixmap.width
    ()image_height = pixmap.height
    image_aspect_ratio = image_width / image_height

    Determine how to scale the image based on its aspect ratio #
    :if image_aspect_ratio <= 0.5
    scaled_pixmap = pixmap.scaledToWidth(label_width,
        (Qt.SmoothTransformation
    :elif image_aspect_ratio >= 1.5
    scaled_pixmap = pixmap.scaledToHeight(label_height,
        (Qt.SmoothTransformation
    :else
    scaled_pixmap = pixmap.scaled(label_size, Qt.KeepAspectRatio,
        (Qt.SmoothTransformation

```

```

        Set the scaled pixmap to the label #
        (label.setPixmap(scaled_pixmap
        (label.setAlignment(Qt.AlignCenter

:(def set_icon_to_circular_label(label, icon_path, width=None, height=None
    Create QIcon object from the provided icon path #
    (icon = QIcon(icon_path

        Get the size of the icon #
        [icon_size = icon.availableSizes()[0
        ()icon_width = width if width is not None else icon_size.width
        ()icon_height = height if height is not None else icon_size.height

        Load the icon pixmap #
        (pixmap = icon.pixmap(icon_width, icon_height

    Create a transparent QImage with the same size as the icon #
    (image = QImage(pixmap.size(), QImage.Format_ARGB32
        (image.fill(Qt.transparent

        Create a QPainter to draw on the image #
        (painter = QPainter(image
        (painter.setRenderHint(QPainter.Antialiasing

        Create a circular path #
        ()path = QPainterPath
        (()path.addEllipse(image.rect

```

```
Set the painter to use the circular path as a clipping path #
    (painter.setClipPath(path
```

```
Draw the icon onto the circular area #
    (painter.drawPixmap(0, 0, pixmap
```

```
End painting #
    ())painter.end
```

```
Set the circular icon to the label #
    ((label.setPixmap(QPixmap.fromImage(image
```

```
:(def set_button_icon(button, icon_path, width, height
    :try
        (icon = QIcon(icon_path
        (button.setIcon(icon
        (icon_size = QSize(width, height
```

```
Use the provided width and height to scale the icon #
    (scaled_size = icon.pixmap(icon_size).size
    (button.setIconSize(scaled_size
    :except Exception as e
    ("{print(f"Error in setting button icon: {e
```

```
:(def calculate_font_size(text
```

You can adjust the coefficients for the linear relationship #

base_size = 28

reduction_factor = 1

return max(base_size - reduction_factor * len(text), 10) # Ensure the minimum
font size is 10

:(def filter_and_sort_chats(search_str, chat_list

Check if chat_list is a list of tuples or a list of strings #

:if len(chat_list) == 0

[] return

:(if isinstance(chat_list[0], tuple

Filter out tuples where chat_name does not contain the search_str #

:if not search_str

return chat_list

filtered_chats = [(chat_name, unread_messages) for chat_name,
unread_messages in chat_list if

[()]search_str.lower() in chat_name.lower

Sort the filtered_chats based on relevance to the search_str #

,sorted_chats = sorted(filtered_chats

key=lambda x: (not x[0].lower().startswith(search_str.lower()),
(((x[0].lower

return sorted_chats

:(elif isinstance(chat_list[0], str

Filter out strings that do not contain the search_str #

:if not search_str

return chat_list

```
filtered_chats = [chat_name for chat_name in chat_list if search_str.lower()
                  [()]in chat_name.lower
```

```

Sort the filtered_chats based on relevance to the search_str #
    ,sorted_chats = sorted(filtered_chats
key=lambda x: (not x.lower().startswith(search_str.lower())),
                (((()x.lower
                return sorted_chats
                :else
Handle other cases or raise an exception based on your requirements #
    ("raise ValueError("Invalid format for chat_list
```

```

:(def calculate_division_value(friends_list_length
    :if friends_list_length == 0
        return 0
    :elif 1 <= friends_list_length <= 5
        return 1
    :else
        :if friends_list_length % 5 == 0
            return friends_list_length // 5
        return (friends_list_length // 5) + 1
```

```

:(def gets_group_attributes_from_format(group_format
    :if "(" not in group_format
        return group_format, None
    :else
        ("(")parts = group_format.split
```

```

        [id = parts[0][1
        [name = parts[1
        (return name, int(id

:(class ChatBox(QWidget

:(def __init__(self, messages_list, Network, parent=None
        ()__super().__init
        :try

        ()screen = QDesktopWidget().screenGeometry
        Extract the screen width and height #
        ()self.screen_width = screen.width
        ()self.screen_height = screen.height
        self.Network = Network
        self.parent = parent
        self.is_getting_called = self.parent.is_getting_called
        (self.square_label = QLabel(self
        (self.width_of_chat_box = int(self.screen_width * 0.416
        (self.height_of_chat_box = int(self.screen_height * 0.926
        (self.file_dialog = QFileDialog(self
        (self.file_dialog.setFileMode(QFileDialog.ExistingFile
        "(*) filter_str = "All files
        (self.file_dialog.setNameFilter(filter_str
        "" = self.file_name
        (self.image_height = int(self.screen_height * 0.277
        (self.image_width = int(self.screen_width * 0.1197
        [] = self.chats_buttons_list

```



```

        [] = self.border_labels
        """ = self.buttons_style_sheet
        } QPushButton
        ;color: white
        ;font-size: 16px
        /* background-color: rgba(0, 0, 0, 0); /* Transparent background
border: 2px solid #2980b9; /* Use a slightly darker shade for the border
                                                                    /*
                                                                    ;border-radius: 5px
                                                                    {
                                                                    } QPushButton: hover
                                                                    ;background-color: #2980b9
                                                                    {
                                                                    """
        """ = self.call_button_style_sheet
        } QPushButton
        ;color: white
        ;font-size: 16px
        /* background-color: rgba(0, 0, 0, 0); /* Transparent background
                                                                    {
                                                                    } QPushButton: hover
        /* filter: brightness(80%); /* Adjust the brightness to make it darker
                                                                    {
                                                                    """

```

```

(self.draw_message_start_x = int(self.screen_width * 0.323
(self.friends_button_height = int(self.screen_height * 0.0462

```

```

(self.create_group_open = QPushButton(self

```

```

        Load an image and set it as the button's icon #
        ("icon = QIcon("discord_app_assets/add_image_button.png
            (self.create_group_open.setIcon(icon

        Set the desired size for the button #

        width, height = int(self.screen_width * 0.013), int(self.screen_height *
                                                                    (0.023
        button_size = QSize(width, height) # Adjust this to your desired button
                                                                    size
        (self.create_group_open.setFixedSize(button_size

        Scale the icon while keeping its aspect ratio #
        icon_size = QSize(width, height) # Set your desired icon size
        ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
        (scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio

        Set the scaled icon size for the button #
        (self.create_group_open.setIconSize(scaled_size
        """)self.create_group_open.setStyleSheet
            } QPushButton
            ;background-color: transparent
            {
            } QPushButton:pressed
            ;background-color: #202225
            ;border-color: #72767d
            {
            (""

```

```

(self.create_group_open_x = int(self.screen_width * 0.2968
(self.create_group_open_y = int(self.screen_height * 0.132
self.create_group_open.move(self.create_group_open_x,
                             (self.create_group_open_y
(self.create_group_open.clicked.connect(self.create_group_clicked
                                     (self.text_entry = QLineEdit(self
                                     ())self.text_entry.hide

        (self.square_pos = (int(self.screen_width * 0.3125), 0
self.square_label.setGeometry(self.square_pos[0], self.square_pos[1],
                             ,self.width_of_chat_box
                             (self.height_of_chat_box
self.square_label.setStyleSheet(f"background-color:
                                {self.parent.background_color_hex}; border: 5px solid
                                (";{{self.parent.standard_hover_color

        [around_name_y = self.square_pos[1]
        [around_name_x = self.square_pos[0]
        (self.around_name = QLabel(self
self.around_name.setStyleSheet(f"background-color:
                                {self.parent.background_color_hex}; border: 5px solid
                                (";{{self.parent.standard_hover_color

(start_height_of_around_name = int(self.screen_height * 0.0462
height_of_around_name = start_height_of_around_name
(self.around_name_delta = int(self.screen_height * 0.2037
    if (self.parent.is_calling and self.parent.selected_chat ==
        \ self.parent.calling_to) or
self.parent.is_in_a_call and self.parent.selected_chat ==)
        :(self.parent.in_call_with
height_of_around_name = start_height_of_around_name +
                             self.around_name_delta

```

```

        [] = self.call_profiles_list

        self.current_chat, self.current_group_id =
        (gets_group_attributes_from_format(self.parent.selected_chat
        :if self.current_group_id
        :(if self.parent.is_call_dict_exist_by_group_id(self.current_group_id
        height_of_around_name = start_height_of_around_name +
        self.around_name_delta

self.around_name.setGeometry(self.square_pos[0], around_name_y,
        (self.width_of_chat_box, height_of_around_name
        (self.around_name.move(around_name_x, around_name_y
        ())_self.around_name.raise

        [] = self.call_profiles_list

        :"" !=! if self.parent.selected_chat
temp_widget_x, temp_widget_y = (int(self.screen_width * 0.3177),
        (height_of_around_name
        (temp_widget_width = int(self.width_of_chat_box * 0.98
        :if height_of_around_name != start_height_of_around_name
        temp_widget_height = self.height_of_chat_box -
        int(self.screen_height * 0.12037) - self.around_name_delta
        :else
        temp_widget_height = self.height_of_chat_box -
        (int(self.screen_height * 0.12037

        if self.parent.is_messages_need_update or
        :self.parent.messages_content_saver is None

```

```

temp_widget = MessageBox(self, temp_widget_width,
                           (temp_widget_height, temp_widget_x, temp_widget_y
self.parent.messages_content_saver = temp_widget
                           self.parent.is_messages_need_update = False
                           :else

(self.parent.messages_content_saver.update_scroll_area_parent(self
                           ()_self.around_name.raise
                           (self.ringing_square_label = QLabel(self

                           (self.send_image_button = QPushButton(self

Load an image and set it as the button's icon #
(icon = QIcon("discord_app_assets/add_image_button.png
                           (self.send_image_button.setIcon(icon

Set the desired size for the button #
width, height = int(self.screen_width * 0.018), int(self.screen_height *
                                                                    (0.032

button_size = QSize(width, height) # Adjust this to your desired button
                                                                    size

                           (self.send_image_button.setFixedSize(button_size

Scale the icon while keeping its aspect ratio #
icon_size = QSize(width, height) # Set your desired icon size
                           ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
(scaled_size = icon_actual_size.scaled(icon_size, Qt.KeepAspectRatio

Set the scaled icon size for the button #

```

```

        (self.send_image_button.setIconSize(scaled_size
        (self.send_image_y = int(self.screen_height * 0.859
        (self.send_image_x = int(self.screen_width * 0.3177
(self.send_image_button.move(self.send_image_x, self.send_image_y
        """self.send_image_button.setStyleSheet(f
            }} QPushButton: hover
;{background-color: {self.parent.standard_hover_color
            {{
            }} QPushButton
;background-color: transparent
            {{
            }} QPushButton: pressed
;background-color: #202225
;border-color: #72767d
            {{
            ("""
(self.send_image_button.clicked.connect(self.open_file_dialog

:try
if self.parent.is_calling and self.parent.selected_chat ==
: self.parent.calling_to
:try
(y_of_label = int(self.screen_height * 0.087
(rings_to_x = int(self.screen_width * 0.479
:(""))if self.parent.selected_chat.startswith
"...text = f"Ringing Group
:else
"...text = f"Ringing User
(self.ringing_to_label = QLabel(text, self

```

```

self.ringing_to_label.setStyleSheet("color: gray; font-size: 14px;
                                     (";margin: 10px

    (self.ringing_to_label.move(rings_to_x, y_of_label
        )text_1_width = self.ringing_to_label.width

        "{text = f'{self.parent.calling_to
        (self.calling_to_label = QLabel(text, self
self.calling_to_label.setStyleSheet("color: white; font-size:
                                     (";25px; margin: 10px

        )text_2_width = self.calling_to_label.width
self.calling_to_label.move(rings_to_x + (text_1_width -
                                     (text_2_width), y_of_label + 20

    (self.stop_calling_button = QPushButton(self

        Set button styles #
        width, height = int(self.screen_width * 0.026),
                           (int(self.screen_width * 0.026
button_size = QSize(width, height) # Adjust this to your desired
                                     button size

    (self.stop_calling_button.setFixedSize(button_size

    ("icon = QIcon("discord_app_assets/reject_button.png
        (self.stop_calling_button.setIcon(icon
        width, height = int(self.screen_width * 0.033),
                           (int(self.screen_width * 0.033

    icon_size = QSize(width, height) # Set your desired icon size
    ([icon_actual_size = icon.actualSize(icon.availableSizes())[0

```

```

scaled_size = icon_actual_size.scaled(icon_size,
                                       (Qt.KeepAspectRatio
                                       (self.stop_calling_button.setIconSize(scaled_size
("self.stop_calling_button.setObjectName("stop_calling_button
original_style_sheet = self.buttons_style_sheet

```

Define a more specific style for self.stop_calling_button #

```

        """ = specific_style
    } QPushButton#stop_calling_button
        ;border: none
        {
            """

```

Append the specific style for self.stop_calling_button to the #
original style sheet

```

modified_style_sheet = original_style_sheet + specific_style

```

Apply the modified style sheet to the button #

```

(self.stop_calling_button.setStyleSheet(modified_style_sheet
stop_calling_button_x, stop_calling_button_y = rings_to_x +
    (text_1_width // 2) - int(self.screen_width * 0.008), y_of_label +
    (int(self.screen_height * 0.101
self.stop_calling_button.move(stop_calling_button_x,
    (stop_calling_button_y
(self.stop_calling_button.clicked.connect(self.stop_calling
    :except Exception as e
    ("{{print(f"error in showing calling {e
    :if self.parent.is_getting_called
    (pop_up_x = int(self.screen_width * 0.4427

```



```

        (pop_up_y = int(self.screen_height * 0.231
        (pop_up_width = int(self.screen_width * 0.1041
        (pop_up_height = int(self.screen_height * 0.2777

        "...text = f\"Incoming Call
        (y_of_label = pop_up_y + int(self.screen_height * 0.1111
        (self.incoming_call_label = QLabel(text, self
        self.incoming_call_label.setStyleSheet("color: gray; font-size:
                                                " 14px; margin: 10px
        (";background-color: transparent;\"
        self.incoming_call_label.move(pop_up_x + int(self.screen_width *
                                                (0.0234), y_of_label

        "{text = f\"{self.parent.getting_called_by
        (self.caller_label = QLabel(text, self
        start_point = int(len(text) - 4) * 1
        (font_size = calculate_font_size(text
        :if start_point < 0
        (start_point = pop_up_x + int(self.screen_width * 0.026
        :else
        start_point = pop_up_x + int(self.screen_width * 0.026) -
                                                start_point

        self.caller_label.setStyleSheet(f\"color: white; font-size:
                                                \" ;{font_size}px; margin: 10px
        (\";background-color: transparent\"
        self.caller_label.move(start_point, y_of_label -
        ((int(self.screen_height * 0.02777

        (self.pop_up_label = QLabel(self

```

```

        ("custom_color = QColor("#053d76
self.pop_up_label.setStyleSheet(f"background-color:
                                (";{()}{custom_color.name
self.pop_up_label.setGeometry(pop_up_x, pop_up_y,
                                (pop_up_width, pop_up_height

(self.accept_button = QPushButton(self
    (self.reject_button = QPushButton(self

        Set button styles #
        width, height = int(self.screen_width *
                            (0.0182),int(self.screen_width * 0.0182
button_size = QSize(width, height) # Adjust this to your desired
                                button size

    (self.accept_button.setFixedSize(button_size
    (self.reject_button.setFixedSize(button_size
(Set button icons (assuming you have phone icons available #

    ("icon = QIcon("discord_app_assets/accept_button.png
        (self.accept_button.setIcon(icon

width, height = int(self.screen_width * 0.026), int(self.screen_width
                                (* 0.026

    icon_size = QSize(width, height) # Set your desired icon size
    ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
        scaled_size = icon_actual_size.scaled(icon_size,
                                                (Qt.KeepAspectRatio

    (self.accept_button.setIconSize(scaled_size

```

```

(self.accept_button.setStyleSheet(self.call_button_style_sheet
    ("icon = QIcon("discord_app_assets/reject_button.png
        (self.reject_button.setIcon(icon
icon_size = QSize(width, height) # Set your desired icon size
    ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
        scaled_size = icon_actual_size.scaled(icon_size,
            (Qt.KeepAspectRatio
            (self.reject_button.setIconSize(scaled_size
                Set button positions #
    (accept_button_x = pop_up_x + int(self.screen_width * 0.059
    (reject_button_x = pop_up_x + int(self.screen_width * 0.026

    self.accept_button.move(accept_button_x, pop_up_y +
        ((int(self.screen_height * 0.185
    self.reject_button.move(reject_button_x, pop_up_y +
        ((int(self.screen_height * 0.185
    (self.reject_button.setStyleSheet(self.call_button_style_sheet
        Connect button signals to functions #
    (self.accept_button.clicked.connect(self.accept_call
        (self.reject_button.clicked.connect(self.reject_call
    if self.parent.is_in_a_call and self.parent.selected_chat ==
        :self.parent.in_call_with
        :try
    (share_camera_height = int(self.screen_height * 0.041666
    share_camera_button_width = int(self.screen_height *
        (0.041666
    (share_camera_x = int(self.screen_width * 0.4375
    (share_camera_y = int(self.screen_height * 0.199
        self.share_camera_off_icon =
        ("QIcon("discord_app_assets/no_camera_icon.png

```

```

        self.share_camera_on_icon =
            ("QIcon("discord_app_assets/camera_icon.png

        self.share_camera_button =
            self.create_custom_in_call_button(share_camera_height,
            ,share_camera_button_width, share_camera_x

        share_camera_y,
            (self.share_camera_and_unshare

        (share_screen_height = int(self.screen_height * 0.041666
        share_screen_button_width = int(self.screen_height *
            (0.041666

        (share_screen_x = int(self.screen_width * 0.4713
        (share_screen_y = int(self.screen_height * 0.199

        self.share_screen_button =
            ,self.create_custom_in_call_button(share_screen_height

            ,share_screen_button_width

        ,share_screen_x
        ,share_screen_y

            (self.share_screen_and_unshare

                :try

                :try

                :if self.parent.is_camera_shared
        set_button_icon(self.share_camera_button,
            self.share_camera_on_icon, share_camera_height,
            (share_camera_button_width

                :else

        set_button_icon(self.share_camera_button,
            ,self.share_camera_off_icon, share_camera_height

        (share_camera_button_width

            :except Exception as e

        ("{print(f"error in setting camera icon {e

```

```

        self.share_screen_off_icon =
        ("QIcon("discord_app_assets/share_screen_off_icon.png

        self.share_screen_on_icon =
        ("QIcon("discord_app_assets/share_screen_on_icon.png

        :try

        :if self.parent.is_screen_shared

        set_button_icon(self.share_screen_button,
        (self.share_screen_on_icon, share_screen_height, share_screen_button_width

        :else

        set_button_icon(self.share_screen_button,
        ,self.share_screen_off_icon, share_screen_height

        (share_screen_button_width

        :except Exception as e

        ("{print(f"error in setting icon for share screen button{e

        :except Exception as e

        ("{print(f"error in creating shares buttons {e

(deafen_button_height = int(self.screen_height * 0.041666
(deafen_button_width = int(self.screen_height * 0.041666

        self.deafened_icon =
        ("QIcon("discord_app_assets/deafened.png

        self.not_deafened_icon =
        ("QIcon("discord_app_assets/not_deafened.png

(deafen_x = share_screen_x + int(self.screen_width * 0.0338

        deafen_y = share_screen_y

        self.deafen_button =
self.create_custom_in_call_button(deafen_button_width, deafen_button_height,
        (deafen_x, deafen_y, self.deafen_and_undeafen

        :if self.parent.deafen

```

```

set_button_icon(self.deafen_button, self.deafened_icon,
                (deafen_button_width, deafen_button_height
                :else
set_button_icon(self.deafen_button, self.not_deafened_icon,
                ,deafen_button_width
                (deafen_button_height

```

```

(mic_button_height = int(self.screen_height * 0.041666
(mic_button_width = int(self.screen_height * 0.041666
                self.unmuted_mic_icon =
                ("QIcon("discord_app_assets/mic_not_muted_icon.png
                self.muted_mic_icon =
                ("QIcon("discord_app_assets/mic_muted_icon.png
(mic_x = deafen_x + int(self.screen_width * 0.0338
                mic_button_y = share_screen_y
                self.mic_button =
self.create_custom_in_call_button(mic_button_width, mic_button_height, mic_x,
                (mic_button_y, self.mute_and_unmute
                :if self.parent.mute
set_button_icon(self.mic_button, self.muted_mic_icon,
                (mic_button_width, mic_button_height
                :else
set_button_icon(self.mic_button, self.unmuted_mic_icon,
                (mic_button_width, mic_button_height

```

```

(self.end_call_button = QPushButton(self

```

Set button styles #

```

(call_button_height = int(self.screen_width * 0.0364
(call_button_width = int(self.screen_width * 0.0364

```

```

button_size = QSize(call_button_width, call_button_height) #
                        Adjust this to your desired button size

        (self.end_call_button.setFixedSize(button_size
                        set_button_icon(self.end_call_button,
("discord_app_assets/reject_button.png", call_button_width, call_button_height

        (self.end_call_button.setStyleSheet(self.call_button_style_sheet
(end_call_button_x = mic_x + int(self.screen_width * 0.028
        ,self.end_call_button.move(end_call_button_x
share_screen_y-int(self.screen_height *
                                                    ((0.013888

(self.end_call_button.clicked.connect(self.end_current_call
        ()self.put_call_icons_on_the_screen
                        :except Exception as e
        ("{print(f"error in incall func {e
                        :except Exception as e
        ("{print(f"error in first try in chatbox {e

```

```

        Load an image and set it as the button's icon #
(icon = QIcon("discord_app_assets/ringing_blue_icon.png
        call_button_x = int(self.screen_width * 0.3125) +
        ((self.width_of_chat_box // 2) + int(self.screen_width * 0.1777
        (call_button_y = int(self.screen_height * 0.0074
self.call_button = self.create_top_page_button(call_button_x,
                                                (call_button_y, icon
        (self.call_button.clicked.connect(self.call_user
        ("icon = QIcon("discord_app_assets/add_user.png
(self.add_user_x = call_button_x - int(self.screen_width * 0.026
        self.add_user_y = call_button_y

```

```

self.add_user_button = self.create_top_page_button(self.add_user_x,
                                                    (self.add_user_y, icon

(self.add_user_button.clicked.connect(self.add_user_to_group_pressed

                                                    :if self.current_group_id
                                                    group_manager =
(self.parent.get_group_manager_by_group_id(self.current_group_id
                                                    :if group_manager == self.parent.username
                                                    ("icon = QIcon("discord_app_assets/edit_name.png
rename_group_x = self.add_user_x - int(self.screen_width *
                                                    (0.026
                                                    rename_group_y = call_button_y
                                                    self.rename_group =
(self.create_top_page_button(rename_group_x, rename_group_y, icon
                                                    ("icon = QIcon("discord_app_assets/edit_image_icon.png
edit_group_image_x = rename_group_x - int(self.screen_width *
                                                    (0.026
                                                    edit_group_image_y = rename_group_y
                                                    self.edit_group_image_button =
, self.create_top_page_button(edit_group_image_x, edit_group_image_y
                                                    (icon

(self.edit_group_image_button.clicked.connect(self.change_group_image

if in chat where there is a group call gives option to join #
                                                    :if self.current_group_id
                                                    if
self.parent.is_call_dict_exist_by_group_id(self.current_group_id) and not
                                                    :self.parent.is_in_a_call

(icon_size = int(self.screen_width * 0.03125

```



```

        (y_of_label = int(self.screen_height * 0.0879
        (rings_to_x = int(self.screen_width * 0.4791
        ("icon = QIcon("discord_app_assets/accept_button.png

(join_button_x = rings_to_x + int(self.screen_width * 0.016666
    (join_button_y = y_of_label + int(self.screen_height * 0.101
    (join_button_width_or_height = int(self.screen_width * 0.026

        self.join_call_button =
        self.create_custom_in_call_button(join_button_width_or_height,
            ,join_button_width_or_height

        join_button_x,

            (join_button_y, self.join_call

    set_button_icon(self.join_call_button, icon, icon_size,

        (icon_size

        ()self.put_call_icons_on_the_screen

        :try

    starter_x_of_group_members, starter_y_of_group_members =
        temp_widget_x + self.width_of_chat_box, around_name_y

        group_members =
        (self.parent.get_group_members_by_group_id(self.current_group_id
        :for member in group_members

            pos = (starter_x_of_group_members,
                (starter_y_of_group_members

    (button = self.create_member_button(member, pos
        ()starter_y_of_group_members += button.height

        :except Exception as e

        ("{print(f"error in drawing members {e

        (self.text_entry = QLineEdit(self

        :if self.parent.background_color == "Black and White

            "text_entry_color = "black

```

```

:else
    "text_entry_color = "white
(text_entry_y = self.send_image_y-int(self.screen_height * 0.0046
    (text_entry_x = int(self.screen_width * 0.33854
    (text_entry_height = int(self.screen_height * 0.037
    self.text_entry.setGeometry(text_entry_x, text_entry_y,
    (self.width_of_chat_box-int(self.screen_width * 0.0364), text_entry_height
    self.text_entry.setStyleSheet(f"background-color:
{self.parent.standard_hover_color}; color: {text_entry_color}; padding: 10px;
    (";border: 1px solid #2980b9; border-radius: 5px; font-size: 14px
    ("", "/")text = self.current_chat.replace
    place_holder_text = "Message" + " " + text
    (self.text_entry.setPlaceholderText(place_holder_text
    ()_self.send_image_button.raise
:else
:try
:if self.parent.is_getting_called
    pop_up_x = 850
    pop_up_y = 250
    pop_up_width = 200
    pop_up_height = 300

    "...text = f"Incoming Call
    y_of_label = pop_up_y + 120
    (self.incoming_call_label = QLabel(text, self
self.incoming_call_label.setStyleSheet("color: gray; font-size:
    " 14px; margin: 10px

    (";background-color: transparent;"
    (self.incoming_call_label.move(pop_up_x + 45, y_of_label

```

```

        "{text = f'{self.parent.getting_called_by
        (self.caller_label = QLabel(text, self
        start_point = int(len(text) - 4) * 1
        (font_size = calculate_font_size(text
        :if start_point < 0
        start_point = pop_up_x + 50
        :else
        start_point = pop_up_x + 50 - start_point

self.caller_label.setStyleSheet(f"color: white; font-size:
                                " ;{font_size}px; margin: 10px
                                (";background-color: transparent"
                                (self.caller_label.move(start_point, y_of_label - 30

                                (self.pop_up_label = QLabel(self
                                ("custom_color = QColor("#053d76
                                self.pop_up_label.setStyleSheet(f"background-color:
                                (";{()}{custom_color.name
                                self.pop_up_label.setGeometry(pop_up_x, pop_up_y,
                                (pop_up_width, pop_up_height

                                (self.accept_button = QPushButton(self
                                (self.reject_button = QPushButton(self

                                Set button styles #
button_size = QSize(35, 35) # Adjust this to your desired button
                                size

                                (self.accept_button.setFixedSize(button_size
                                (self.reject_button.setFixedSize(button_size
                                (Set button icons (assuming you have phone icons available #

```

```

(icon = QIcon("discord_app_assets/accept_button.png
            (self.accept_button.setIcon(icon
            icon_size = QSize(50, 50) # Set your desired icon size
            ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
                scaled_size = icon_actual_size.scaled(icon_size,
                                                        (Qt.KeepAspectRatio

            (self.accept_button.setIconSize(scaled_size
(self.accept_button.setStyleSheet(self.call_button_style_sheet
            ("icon = QIcon("discord_app_assets/reject_button.png
            (self.reject_button.setIcon(icon
            icon_size = QSize(50, 50) # Set your desired icon size
            ([icon_actual_size = icon.actualSize(icon.availableSizes())[0
                scaled_size = icon_actual_size.scaled(icon_size,
                                                        (Qt.KeepAspectRatio

            (self.reject_button.setIconSize(scaled_size
                Set button positions #
                accept_button_x = pop_up_x + 115
                reject_button_x = pop_up_x + 50

            (self.accept_button.move(accept_button_x, pop_up_y + 200
            (self.reject_button.move(reject_button_x, pop_up_y + 200
            (self.reject_button.setStyleSheet(self.call_button_style_sheet
                Connect button signals to functions #
            (self.accept_button.clicked.connect(self.accept_call
            (self.reject_button.clicked.connect(self.reject_call
                :except Exception as e
            ("{print(f"error in getting called {e

```

```

List to store message labels #
                                :try
                                :try

self.chat_name_label = QLabel(self.current_chat.replace("/", ""),
                                (self
self.chat_name_label.setStyleSheet("color: white; font-size: 20px;
                                (";margin: 10px

Set a fixed width for the label #
(self.chat_name_label.setFixedWidth(200
chat_name_label_x = 620
(self.chat_name_label.move(chat_name_label_x, 3
self.messages_list = messages_list
[] = self.message_labels

(self.filename_label = QLabel(self
(self.filename_label.move(620, 830
file_name_y = 860
file_name_x = 620
file_name_width = 300
self.filename_label.setGeometry(file_name_x, file_name_y,
file_name_width, 50) # Adjust the size as needed
self.filename_label.setWordWrap(True) # Enable word wrap
()_self.filename_label.raise
)self.filename_label.setStyleSheet
";background-color: #333333; color: white; font-size: 16px"
(
()self.filename_label.hide

```

```

        (self.garbage_button = QPushButton(self
"garbage_icon_path = "discord_app_assets/garbage_icon.png
        (garbage_icon_width, garbage_icon_height = (35, 35
        set_button_icon(self.garbage_button, garbage_icon_path,
            (garbage_icon_width, garbage_icon_height
self.garbage_button.setGeometry(file_name_x + file_name_width -
        (50, file_name_y+8, garbage_icon_width, garbage_icon_height
            ()self.garbage_button.hide
(self.garbage_button.clicked.connect(self.garbage_button_clicked
        (make_q_object_clear(self.garbage_button

        :"" =! if self.parent.file_name
("self.filename_label.setText(self.parent.file_name + " is loaded
            ()self.filename_label.show
            ()self.garbage_button.show
            :else
        (self.filename_label.setText(self.parent.file_name

        (self.image_too_big = QLabel(self
        (self.image_too_big.move(620, 830
self.image_too_big.setGeometry(620, 830, 200, 50) # Adjust the
            size as needed

self.image_too_big.setWordWrap(True) # Enable word wrap
            ()_self.image_too_big.raise
            )self.image_too_big.setStyleSheet
            ";background-color: #333333; color: red; font-size: 16px"
            (
            ("self.image_too_big.setText("Image size it too big
            :if self.parent.size_error_label

```

```

        ()self.image_too_big.show
        :else
        ()self.image_too_big.hide
        :except Exception as e
        ("{print(f"error in drawing filename and image_too_big {e

        :try
        :try
        (self.chats_label = QLabel("DIRECT MESSAGES", self
        ""self.chats_label.setStyleSheet
        ;color: white
        ;font-size: 12px
        ;padding: 5px
        ;margin-bottom: 2px
        ("
        friend_starter_y = 170
        friend_x = 250
        (self.chats_label.move(friend_x + 15, friend_starter_y - 28
        friends_button_y = 90
        friends_button_height = self.friends_button_height
        border_height = 912
        border_width = 350
        info_y = 902
        (self.border_label = QLabel(self
        ""self.border_label.setStyleSheet(f
        ;{border: 2px solid {self.parent.standard_hover_color
        ;border-radius: 5px
        ;padding: 5px

```

```

        ;margin-bottom: 2px
        (""
self.border_label.setGeometry(friend_x, 0, border_width,
                                (border_height
                                )self.border_label.lower

        (self.border_label2 = QLabel(self
        ""self.border_label2.setStyleSheet(f
        ;padding: 5px
        ;margin-bottom: 2px
        border-top: 2px solid
        /* {self.parent.standard_hover_color}; /* Top border
        border-left: 2px solid
        /* {self.parent.standard_hover_color}; /* Left border
        border-right: 2px solid
        /* {self.parent.standard_hover_color}; /* Right border
        (""
(self.border_label2.setGeometry(friend_x, 0, border_width, 170
                                ()self.border_label2.lower

        (find_contact_pos = (260, 20
        (find_contact_size = (320, 40
        (self.find_contact_text_entry = QLineEdit(self
                                :except Exception as e
        ("print(f"error in first part of creating direct messages button
                                :try
        :if self.parent.background_color == "Black and White
            "text_entry_color = "black
                                :else
            "text_entry_color = "white

```



```

self.find_contact_text_entry.setPlaceholderText("Find a
                                                    ("conversation
                                                    :except Exception as e
                                                    ("{print(f"error in if {e
                                                    :try

                                                    )self.find_contact_text_entry.setStyleSheet
f"background-color: {self.parent.standard_hover_color}; color:
{text_entry_color}; padding: 10px; border: 1px solid #2980b9; border-radius: 5px;
                                                    (";font-size: 14px

self.find_contact_text_entry.setGeometry(find_contact_pos[0],
                                                    ,[find_contact_pos[1], find_contact_size[0]
                                                    ([find_contact_size[1]

self.find_contact_text_entry.textChanged.connect(self.on_text_changed_in_cont
                                                    (act_search

(self.friends_button = QPushButton(" Social", self
                                ""self.friends_button.setStyleSheet(f
                                }} QPushButton
                                ;color: white
                                ;font-size: 15px
                                /* border: none; /* Remove the border
                                ;border-radius: 5px
                                ;padding: 5px
                                ;margin-bottom: 2px
                                /* text-align: left; /* Align the text to the left
                                /* alignment: left; /* Align the icon and text to the left
/* padding-left: 10px; /* Adjust the starting position to the right
                                {{

```

```

        }} QPushButton: hover
;{background-color: {self.parent.standard_hover_color
        {{

        }} QPushButton: pressed
;background-color: #202225
;border-color: #72767d
        {{

        ("
icon = QIcon("discord_app_assets/friends_icon.png") # Replace
        with the path to your icon image
        (self.friends_button.setIcon(icon

        Set the position and size of the button #
self.friends_button.setGeometry(friend_x + 5, friends_button_y -
        ,10, border_width - 15
        friends_button_height) # Adjust size as
        needed

        Set the text alignment to show both the icon and text #

        Optional: Adjust the spacing between the icon and text #
self.friends_button.setIconSize(QSize(50, 50)) # Adjust size as
        needed

        (self.friends_button.clicked.connect(self.parent.social_clicked
        :except Exception as e
        ("{print(f"error in second part of try {e
        :except Exception as e

```

```

print(f"error in creating direct messages button and friends_button
({{e

:try

friend_x = 250

:if not self.parent.current_chat_box_search
(chats_widget = FriendsChatListWidget(self, self.parent.chats_list

:else
chats_widget = FriendsChatListWidget(self,
(self.parent.temp_search_list

:except Exception as e
({{print(f"error in showing chats list{e

:try

(username_label = QLabel(self.parent.username, self

:"if text_entry_color == "black
"username_label_background_color = "black

:else
username_label_background_color =
self.parent.standard_hover_color

username_label_margin_bottom = int(self.parent.screen_width *
(0.013

(username_label_padding = int(self.parent.screen_width * 0.035
"username_label.setStyleSheet(f

;color: white

;font-size: 20px

;{background-color: {username_label_background_color
border: 2px solid {self.parent.standard_hover_color}; /* Use a
/* slightly darker shade for the border

```

```

        ;border-radius: 5px
        ;padding: {username_label_padding}px
        ;margin-bottom: {username_label_margin_bottom}px
    """

    (username_label.setGeometry(friend_x, info_y, border_width, 90

        profile_image_label_position = int(friend_x +
(self.parent.screen_width * 0.005), int(info_y + self.parent.screen_height * 0.004
        (width, height = (55, 55
    profile_image_label = create_custom_circular_label(width, height,
                                                (self
                                                chat_image =
        (self.parent.get_profile_pic_by_username(self.parent.username
                                                :if chat_image is None
        icon_path = self.parent.regular_profile_image_path
    (set_icon_from_path_to_label(profile_image_label, icon_path
                                                :else
        circular_pic_bytes =
    (self.parent.get_circular_image_bytes_by_name(self.parent.username
        set_icon_from_bytes_to_label(profile_image_label,
                                                (circular_pic_bytes
    profile_image_label.move(profile_image_label_position[0],
        ([profile_image_label_position[1
        :except Exception as e
    ("{print(f"error in drawing profile pic in chatbox {e

    (settings_button = QPushButton(self
settings_button.setFixedSize(50, 50) # Set the fixed size of the button
        Set the icon for the chat button #

```

```

        settings_button_icon =
        ("QIcon(QPixmap("discord_app_assets/Setting_logo.png
        (settings_button.setIcon(settings_button_icon
settings_button.setIconSize(settings_button_icon.actualSize(QSize(50,
        50))) # Adjust the size as needed
        (settings_button.move(friend_x + 295, info_y + 10
        ""))settings_button.setStyleSheet
        } QPushButton
        ;background-color: transparent
        {

        ("
        (settings_button.clicked.connect(self.parent.settings_clicked
        (pause_mp3_files_button = QPushButton(self
        "mp3_pause_path = "discord_app_assets/pause_and_play_icon.png
        (set_button_icon(pause_mp3_files_button, mp3_pause_path, 40, 40
        (pause_mp3_files_button.move(friend_x + 240, info_y + 10
        ""))pause_mp3_files_button.setStyleSheet
        } QPushButton
        ;background-color: transparent
        {

        ("

pause_mp3_files_button.clicked.connect(self.parent.pause_or_unpause_mp3_fil
        (es_player

        (music_page_button = QPushButton(self
        "mp3_pause_path = "discord_app_assets/music_icon.png
        (set_button_icon(music_page_button, mp3_pause_path, 40, 40

```

```

(music_page_button.move(friend_x + 185, info_y + 10
                        ""))music_page_button.setStyleSheet
                        } QPushButton
                        ;background-color: transparent
                        {

                        ("
(music_page_button.clicked.connect(self.parent.music_button_clicked
                                :try
                                :if self.parent.is_create_group_pressed
                                create_group_box = CreateGroupBox(self,
                                ("self.create_group_open_x, self.create_group_open_y, "create
                                ()_create_group_box.raise
                                :elif self.parent.is_create_group_inside_chat_pressed
                                :if self.parent.is_current_chat_a_group
                                create_group_box = CreateGroupBox(self, self.add_user_x,
                                ("self.add_user_y, "add
                                :else
                                create_group_box = CreateGroupBox(self, self.add_user_x,
                                ("self.add_user_y, "create
                                ()_create_group_box.raise
                                ()_chats_widget.raise
                                ()self.raise_needed_elements
                                :except Exception as e
                                ("{print(f"error in creaing CreateGroupBox in chatbox {e
                                :except Exception as e
                                ("{print(f"error in last level in creating chat box {e
                                :except Exception as e
                                ("{print(f"error in creating chat box {e

```

```

:(def add_user_to_group_pressed(self
  :if self.parent.is_create_group_inside_chat_pressed
self.parent.is_create_group_inside_chat_pressed = False
  ()self.parent.update_chat_page_without_messages
  :else
    self.parent.is_create_group_pressed = False
self.parent.is_create_group_inside_chat_pressed = True
  ()self.parent.update_chat_page_without_messages

```

```

:(def change_group_image(self
  ()self.open_file_dialog_for_changing_group_image

```

Layout #

```

:(def create_custom_in_call_button(self, width, height, x, y, click_function
  (button = QPushButton(self

  (button_size = QSize(width, height
  (button.setFixedSize(button_size

  (button.move(x, y

  (button.clicked.connect(click_function

  ""button.setStyleSheet(f
    }} QPushButton
    ;background-color: #6fa8b6
    ;background-repeat: no-repeat

```

```

;background-position: center
/* border-radius: {height // 2}px; /* Set to half of the button height
}}
}} QPushButton: hover
;{background-color: {self.parent.standard_hover_color
}}
("""

return button

:(def put_call_icons_on_the_screen(self
:try
:if self.current_group_id
current_call_dict =
(self.parent.get_call_dict_by_group_id(self.current_group_id
("{print(f'dict is {current_call_dict
:else
current_call_dict =
(self.parent.get_call_dict_by_user(self.parent.username
("{print(f'dict is {current_call_dict
(("numbers_of_users_in_call = len(current_call_dict.get("participants
(starts_x = 900+((numbers_of_users_in_call-2) * -70
y_of_profiles = 95

:if current_call_dict is not None
("names = current_call_dict.get("participants
:if self.parent.username in names
:for name in names

```



```

        self.create_profile_button(starts_x, y_of_profiles, name,
                                   (current_call_dict
                                   if name in current_call_dict.get("screen_streamers") and name !=
                                   :self.parent.username

                                   "stream_type = "ScreenStream
self.create_watch_stream_button(starts_x+10, y_of_profiles-35,
                                (name, stream_type
                                if name in current_call_dict.get("camera_streamers") and name !=
                                :self.parent.username

                                "stream_type = "CameraStream
self.create_watch_stream_button(starts_x+10, y_of_profiles-35,
                                (name, stream_type
                                starts_x += 105
                                :except Exception as e
                                ("print(f"error is {e} in icon management

:(def create_watch_stream_button(self, x, y, name, stream_type
    (width, height = (70, 30
    :if stream_type == "ScreenStream
        (button = QPushButton(self
        (button_size = QSize(width, height
        (button.setFixedSize(button_size
        "image_icon = f"discord_app_assets/monitor_icon.png
set_button_icon(button, image_icon, width, height) # Corrected function
                                                    call
                                                    :else
                                                    y -= 50
                                                    (button = QPushButton(self
                                                    (button_size = QSize(width, height
                                                    (button.setFixedSize(button_size

```

```

        "image_icon = "discord_app_assets/camera_watch_icon.png
set_button_icon(button, image_icon, width, height) # Corrected function call

        """button.setStyleSheet(f
            }} QPushButton
        ;{background-color: {self.parent.standard_hover_color
            /* color: white; /* Default font color
            /* border-radius: 15px; /* Adjust the radius as needed
            {{
            }} QPushButton: hover
        ;background-color: #2980b9
            {{
            ("""
        (button.move(x, y

button.clicked.connect(lambda: self.watch_stream_button_pressed(name,
                                                                    ((stream_type
                                                                    (self.call_profiles_list.append(button

:(def watch_stream_button_pressed(self, name, stream_type
    :try
        :if not self.parent.is_watching_screen
        self.parent.is_watching_screen = True
        self.parent.watching_user = name
        self.parent.watching_type = stream_type
        :if stream_type == "ScreenStream
        (self.Network.watch_screen_stream_of_user(name
        :else
        (self.Network.watch_camera_stream_of_user(name

```

```

({print(f"Started watching stream of {name} of type: {stream_type
    )self.parent.start_watching_video_stream
                                :else
    ("print("does not suppose to happen
                                :except Exception as e
    ("{print(f"Problem with watch button, error {e

:(def create_profile_button(self, x, y, name, dict
    (width, height = (90, 90
(button = create_custom_circular_label(width, height, self

    (status_button = QPushButton(self
    (make_q_object_clear(status_button
    (width, height = (30, 30
    (button_size = QSize(width, height
    (status_button.setFixedSize(button_size

    (button.move(x, y

"regular_icon_path = r"discord_app_assets/regular_profile.png
("muted_icon = QIcon("discord_app_assets/mic_muted_icon.png
    ("deafened_icon = QIcon("discord_app_assets/deafened.png
    (regular_icon = QIcon(regular_icon_path
    ("deafened = dict.get("deafened
    ("muted = dict.get("muted

:("if name in dict.get("deafened
(set_button_icon(status_button, deafened_icon, width, height

```

```

        :("elif name in dict.get("muted
(set_button_icon(status_button, muted_icon, width, height

(profile_pic = self.parent.get_circular_image_bytes_by_name(name
        :try
        :if profile_pic is not None
        (set_icon_from_bytes_to_label(button, profile_pic
        :else
        (regular_icon_bytes = file_to_bytes(regular_icon_path
(set_icon_from_bytes_to_label(button, regular_icon_bytes
        :except Exception as e
        ("{print(f"error in setting image to profile button {e
status_button.move(x + int(0.7 * button.width()), y + int(0.7 *
        (((button.height

        (self.call_profiles_list.append(button
        (self.call_profiles_list.append(status_button
        return button

:(def create_top_page_button(self, x, y, icon_path
        (button = QPushButton(self
        (width, height = (35, 35
        (button_size = QSize(width, height
        (button.setFixedSize(button_size

        (button.move(x, y
        """)button.setStyleSheet
        } QPushButton
;background-color: transparent
;background-repeat: no-repeat

```

```

;background-position: center
border-radius: "" + str(height // 2) + ""px; /* Set to half of the button
                                                    /* height
                                                    {
                                                    } QPushButton: hover
;background-color: #2980b9
                                                    {
                                                    (""
(set_button_icon(button, icon_path, width, height
return button

```

```

:(def stop_calling(self
()self.Network.stop_ringing_to_group_or_user

:(def create_group_clicked(self
:if self.parent.is_create_group_pressed
self.parent.is_create_group_pressed = False
()self.parent.selected_group_members.clear
self.parent.create_group_index = 0
:else
self.parent.is_create_group_pressed = True
()self.parent.update_chat_page_without_messages

```

Create Group button #

```

:(def toggle_checkbox(self, create_or_add_group_widget
()sender = self.sender
:(if isinstance(sender, QPushButton
friend_name = sender.friend_name

```

```

        )friend_checkbox = next
        ()child for child in sender.parent().children
if isinstance(child, QCheckBox) and child.friend_name == friend_name
        (
        ()friend_checkbox.toggle
        ()create_or_add_group_widget.update_labels_text
        ((self.friend_checkbox_changed(friend_checkbox.isChecked

        :(def handle_create_group_index(self, format
        change = False
        :try
        :if format == "down
if len(self.parent.friends_list) > (self.parent.create_group_index + 1) *
:5

        self.parent.create_group_index += 1
        change = True
        :else
        :if self.parent.create_group_index > 0
        self.parent.create_group_index -= 1
        change = True
        :except Exception as e
        ("print("error in hadnling index
        :if change
        ()self.parent.update_chat_page_without_messages
        :else
        ("print("no change

        :(def create_dm_pressed(self
        :if len(self.parent.selected_group_members) != 0

```

```

(self.Network.create_group(self.parent.selected_group_members
    ("print("You a created new group
        self.parent.is_create_group_pressed = False
self.parent.is_create_group_inside_chat_pressed = False
    ()self.parent.selected_group_members.clear
        self.parent.create_group_index = 0
    ()self.parent.update_chat_page_without_messages

        :(def add_users_to_group(self
            group_id = self.current_group_id
        :if len(self.parent.selected_group_members) != 0
            self.Network.add_user_to_group(group_id,
                (self.parent.selected_group_members
print(f"Added user {self.parent.selected_group_members} to group of id
    ("{{group_id
        self.parent.is_create_group_pressed = False
self.parent.is_create_group_inside_chat_pressed = False
    ()self.parent.selected_group_members.clear
        self.parent.create_group_index = 0
    ()self.parent.update_chat_page_without_messages

        :(def friend_checkbox_changed(self, state
            ()checkbox = self.sender
            friend_name = checkbox.friend_name
            wanted_len = self.parent.group_max_members - 1
            :try
if state == 2 and len(self.parent.selected_group_members) < wanted_len:
    # Checked state
    (self.parent.selected_group_members.append(friend_name

```

```

:else
:if friend_name in self.parent.selected_group_members and state == 0
    (self.parent.selected_group_members.remove(friend_name
:except Exception as e
    ("{{print(f'friend_checkbox_changed error :{e

:
:(def is_mouse_on_chats_list(self, mouse_pos
    ()box_geometry = self.border_label.geometry
    (return box_geometry.contains(mouse_pos

:
:(def on_text_changed_in_contact_search(self
This function will be called when the text inside QLineEdit changes #
:try
:
:()if self.find_contact_text_entry.hasFocus
:if len(self.find_contact_text_entry.text()) > 0
    self.parent.current_chat_box_search = True
    ()self.parent.temp_search_list = self.return_search_list
    ()self.parent.update_chat_page_without_messages
:else
:try
    self.parent.current_chat_box_search = False
    [] = self.parent.temp_search_list
    ()self.parent.update_chat_page_without_messages
:except Exception as e
    ("{{print(f'text_changed error :{e
:except Exception as e
    ("{{print(f'text_changed error :{e

```



```

:(def return_search_list(self
    Get the filtered and sorted list of buttons #
    :try
filtered_buttons = filter_and_sort_chats(self.find_contact_text_entry.text(),
                                         (self.parent.chats_list

    Remove all existing buttons from the layout #
    Create and add the updated buttons to the layout #
    [] = temp_list
    :(if not isinstance(filtered_buttons[0], str
    :for chat_name, button in filtered_buttons
        (temp_list.append(chat_name
            return temp_list
        :else
    :for chat_name in filtered_buttons
        (temp_list.append(chat_name
            return temp_list
    :except Exception as e
    ("{print(f"return_search_list error :{e

:(def raise_needed_elements(self
    :try
    :"" =! if self.parent.selected_chat
    ()_self.add_user_button.raise
    :if self.current_group_id
        group_manager =
    (self.parent.get_group_manager_by_group_id(self.current_group_id
    :if group_manager == self.parent.username
        ()_self.rename_group.raise
        ()_self.edit_group_image_button.raise

```

```

        ()_self.border_label2.raise
    ()_self.find_contact_text_entry.raise
        ()_self.friends_button.raise
            ()_self.chats_label.raise
        ()_self.create_group_open.raise
            :if self.parent.is_calling
                ()_self.ringing_to_label.raise
                ()_self.calling_to_label.raise
            ()_self.stop_calling_button.raise
                :if self.parent.is_getting_called
                    ()_self.pop_up_label.raise
                    ()_self.accept_button.raise
                    ()_self.reject_button.raise
                ()_self.incoming_call_label.raise
                    ()_self.caller_label.raise
                        :if self.parent.is_in_a_call
                            ()_self.mic_button.raise
                            ()_self.end_call_button.raise
                        ()_self.share_screen_button.raise
                            ()_self.deafen_button.raise
                        ()_self.share_camera_button.raise
                    :for profile_button in self.call_profiles_list
                        ()_profile_button.raise
                            :if self.current_group_id
                                if self.parent.is_call_dict_exist_by_group_id(self.current_group_id) and
                                    :not self.parent.is_in_a_call
                                        ()_self.join_call_button.raise
                                    :for profile_button in self.call_profiles_list
                                        ()_profile_button.raise

```

```

                                :except Exception as e
                                ("{print(f"error in raising elements {e

:(def create_friend_button(self, label, position

                                px_padding_of_button_text = 55
                                chat_name = label
                                (text, id = gets_group_attributes_from_format(chat_name
                                :if id
                                (len_group = self.parent.get_number_of_members_by_group_id(id
                                button_text = text

                                (width, height = (35, 35
                                (profile_image_label = create_custom_circular_label(width, height, self
                                profile_image_x, profile_image_y = (position[0] +
                                (px_padding_of_button_text * 0.25), position[1] + ((self.friends_button_height -
                                ((height) * 0.5

                                :if id
                                (chat_image = self.parent.get_circular_image_bytes_by_group_id(id
                                :else
                                (chat_image = self.parent.get_profile_pic_by_username(chat_name
                                :if chat_image is None
                                icon_path = self.parent.regular_profile_image_path
                                (set_icon_from_path_to_label(profile_image_label, icon_path
                                :else
                                :if id
                                circular_pic_bytes = chat_image
                                :else

```

```

        circular_pic_bytes =
            (self.parent.get_circular_image_bytes_by_name(chat_name
(set_icon_from_bytes_to_label(profile_image_label, circular_pic_bytes
            (profile_image_label.move(profile_image_x, profile_image_y

        (button = QPushButton(self
            (button.setText(button_text
                ([button.move(position[0], position[1
                    (button.setFixedHeight(self.friends_button_height
                ((button.clicked.connect(partial(self.on_friend_button_clicked, label
                    (button.setContextMenuPolicy(Qt.CustomContextMenu
                        :if not id
                            ["actions_list = ["remove_chat
                                )button.customContextMenuRequested.connect
                                    ,lambda pos, parent=self, button=button, actions_list=actions_list
chat_name=chat_name: self.parent.right_click_object_func(pos,
                                                                ,parent, button
                    ((actions_list, chat_name

                        :else
                            ["actions_list = ["exit_group
                                )button.customContextMenuRequested.connect
                                    ,lambda pos, parent=self, button=button, actions_list=actions_list
                    ,group_id=id: self.parent.right_click_object_func(pos, parent, button
                    ((actions_list, group_id=group_id

        "" = style
        ;color: white
        ;font-size: 10px
        ;margin-bottom: 2px

```

```

;(background-color: rgba(0,0,0,0
    ""

    :if id

    (members_label = QLabel(f"{len_group} Members", self
        (members_label.setStyleSheet(style
    memeber_x = position[0] + px_padding_of_button_text
        (members_label.move(memeber_x, position[1] + 28

padding_top = "padding-top: -7px;" if label.startswith("(") else "" # Adjust the
                                padding value as needed

        ""button.setStyleSheet(f
                                }} QPushButton
                                ;{background-color: {self.parent.background_color_hex
                                ;{border: 2px solid {self.parent.standard_hover_color
                                ;border-radius: 5px
                                ;padding: 8px 16px
padding-left: {px_padding_of_button_text}px; /* Adjust the padding to
                                                /* move text to the right
                                {padding_top}
                                ;color: white
                                ;font-family: Arial, sans-serif
                                ;font-size: 14px
                                ;font-weight: normal
                                ;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
                                ;text-align: left
                                {{

```

```

        }} QPushButton: hover
;{background-color: {self.parent.standard_hover_color
        {{

        }} QPushButton: pressed
;background-color: #202225
;border-color: #72767d
        {{
        (""

(button.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Fixed
        (button.setFixedWidth(350
                ()_button.raise
        ()_profile_image_label.raise
                :if id
        ()_members_label.raise

        return button

:(def create_member_button(self, label, position

        px_padding_of_button_text = 55
        chat_name = label
        button_text = chat_name

        (width, height = (35, 35
(profile_image_label = create_custom_circular_label(width, height, self

```

```

        profile_image_x, profile_image_y = (position[0] +
(px_padding_of_button_text * 0.25), position[1] + ((self.friends_button_height -
                                                    ((height) * 0.5

```

```

(chat_image = self.parent.get_profile_pic_by_username(chat_name
                                                    :if chat_image is None
        icon_path = self.parent.regular_profile_image_path
        (set_icon_from_path_to_label(profile_image_label, icon_path
                                                    :else
        circular_pic_bytes =
        (self.parent.get_circular_image_bytes_by_name(chat_name
        (set_icon_from_bytes_to_label(profile_image_label, circular_pic_bytes
        (profile_image_label.move(profile_image_x, profile_image_y
        (button = QPushButton(self
        (button.setText(button_text
        ([button.move(position[0], position[1
        (button.setFixedHeight(self.friends_button_height
        (button.setContextMenuPolicy(Qt.CustomContextMenu
        manager =
        (self.parent.get_group_manager_by_group_id(self.current_group_id
        :if chat_name != self.parent.username
        :if chat_name not in self.parent.friends_list
        ["actions_list = ["add_friend", "message_user
        :if manager == self.parent.username
        ("actions_list.append("remove_user_from_group
        )button.customContextMenuRequested.connect
        ,lambda pos, parent=self, button=button, actions_list=actions_list
        chat_name=chat_name, group_id=self.current_group_id:
        ,self.parent.right_click_object_func(pos, parent, button

```

```

actions_list, chat_name,
                                ((group_id
                                :else

                                ["actions_list = ["message_user
                                :if manager == self.parent.username
                                ("actions_list.append("remove_user_from_group
                                )button.customContextMenuRequested.connect
                                ,lambda pos, parent=self, button=button, actions_list=actions_list
                                chat_name=chat_name, group_id=self.current_group_id:
                                ,self.parent.right_click_object_func(pos, parent, button
                                actions_list, chat_name,
                                ((group_id

padding_top = "padding-top: -7px;" if label.startswith("(") else "" # Adjust the
                                padding value as needed
                                """"button.setStyleSheet(f
                                }} QPushButton
                                ;{background-color: {self.parent.background_color_hex
                                ;{border: 2px solid {self.parent.standard_hover_color
                                ;border-radius: 5px
                                ;padding: 8px 16px
                                padding-left: {px_padding_of_button_text}px; /* Adjust the padding to
                                /* move text to the right
                                {padding_top}
                                ;color: white
                                ;font-family: Arial, sans-serif
                                ;font-size: 14px
                                ;font-weight: normal
                                ;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
                                ;text-align: left

```



```

        {{
        ("""

(button.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Fixed
                        (button.setFixedWidth(350
                        ()_button.raise
                        ()_profile_image_label.raise

                        return button

:(def raise_around_name_label(self
    ()_self.around_name.raise

:(def open_file_dialog_for_changing_group_image(self
    :()_if self.file_dialog.exec
    ()selected_files = self.file_dialog.selectedFiles
[file_types = [os.path.splitext(file)[1][1:].lower() for file in selected_files
    file_path = selected_files[0] # Get the file path
    file_size = os.path.getsize(file_path) # Get the file size in bytes

(Check if the file size is greater than 10 MB (10 * 1024 * 1024 bytes #
    :if file_size > 10 * 1024 * 1024
    (".(print("File size exceeds the limit (10 MB
    return

:["if selected_files and file_types[0] in ["png", "jpg
([image_bytes = file_to_bytes(selected_files[0

:(if is_valid_image(image_bytes

```

```

self.Network.send_new_group_image_to_server(image_bytes,
                                             (self.current_group_id
                                              ("print("sent new image to server
                                              :else
                                              ("print("couldn't load image

                                              : (def open_file_dialog(self
["basic_files_types = ["xlsx", "py", "docx", "pptx", "txt", "pdf
                                              : ()_if self.file_dialog.exec
                                              ()selected_files = self.file_dialog.selectedFiles
[file_types = [os.path.splitext(file)[1][1:].lower() for file in selected_files
[ self.parent.file_name = selected_files[0].split("/")[-1
file_path = selected_files[0] # Get the file path
file_size = os.path.getsize(file_path) # Get the file size in bytes

(Check if the file size is greater than 10 MB (10 * 1024 * 1024 bytes #
:if file_size > 10 * 1024 * 1024
(. (print("File size exceeds the limit (10 MB
return
:["if selected_files and file_types[0] in ["png", "jpg
([image_bytes = file_to_bytes(selected_files[0

: (if is_valid_image(image_bytes
self.parent.file_to_send = image_bytes
(print("image to send defined
("self.filename_label.setText(self.parent.file_name + " is loaded
()self.filename_label.show
(self.parent.update_chat_page_without_messages
()self.parent.activateWindow

```

```

:else

    ("print("couldn't load image

:elif selected_files and file_types[0] in ["mp4", "mov
    ([video_bytes = file_to_bytes(selected_files[0
        self.parent.file_to_send = video_bytes
("self.filename_label.setText(self.parent.file_name + " is loaded
        ()self.filename_label.show
    ()self.parent.update_chat_page_without_messages
        ()self.parent.activateWindow
:elif selected_files and file_types[0] in ["mp3
    ([audio_bytes = file_to_bytes(selected_files[0
        self.parent.file_to_send = audio_bytes
("self.filename_label.setText(self.parent.file_name + " is loaded
        ()self.filename_label.show
    ()self.parent.update_chat_page_without_messages
        ()self.parent.activateWindow
:elif selected_files and file_types[0] in basic_files_types
    ([file_bytes = file_to_bytes(selected_files[0
        self.parent.file_to_send = file_bytes
("self.filename_label.setText(self.parent.file_name + " is loaded
        ()self.filename_label.show
    ()self.parent.update_chat_page_without_messages
        ()self.parent.activateWindow

: (def file_to_bytes(self, file_path
: with open(file_path, "rb") as file
    ()image_bytes = file.read
        max_size_kb = 40

```

```

max_size_bytes = max_size_kb * 1024 # Convert KB to bytes
        if len(image_bytes) > max_size_bytes
            ("print("Image size exceeds 40 KB
                ()self.image_too_big.show
            self.parent.size_error_label = True
                return
            :else
                ()self.image_too_big.hide
            self.parent.size_error_label = False
                return image_bytes

        :(def join_call(self
            :try
                ("print("trying to join call of current group
                    self.parent.is_joining_call = True
                self.parent.joining_to = self.parent.selected_chat
            (self.Network.send_join_call_of_group_id(self.current_group_id
                :except Exception as e
                    ("{"print(f"error in joining call {e

        :(def end_current_call(self
            ()self.parent.end_current_call

        :(def mute_and_unmute(self
            :try
                :if self.parent.mute
                    media_content =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Discord_mute_sound_e
                        (('ffect.mp3

```

```

        (self.parent.play_sound_effect(media_content
            ("print("mic is not muted
            self.parent.mute = False
        (self.mic_button.setIcon(self.unmuted_mic_icon
            ()self.Network.toggle_mute_for_myself
            :else
                media_content =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Discord_mute_sound_e
                (('fect.mp3
        (self.parent.play_sound_effect(media_content
            ("print("mic is muted
            self.parent.mute = True
        (self.mic_button.setIcon(self.muted_mic_icon
            ()self.Network.toggle_mute_for_myself
            :except Exception as e
                ("{print(f"error mute_and_unmute {e

        :(def deafen_and_undeafen(self
            :if self.parent.deafen
                media_content =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Discord_mute_sound_e
                (('fect.mp3
        (self.parent.play_sound_effect(media_content
            self.parent.deafen = False
        (self.deafen_button.setIcon(self.not_deafened_icon
            ()self.Network.toggle_deafen_for_myself
            :else
                media_content =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Discord_mute_sound_e
                (('fect.mp3

```

```

        (self.parent.play_sound_effect(media_content
                                     self.parent.deafen = True
        (self.deafen_button.setIcon(self.deafened_icon
                                     ()self.Network.toggle_deafen_for_myself

        :(def share_camera_and_unshare(self
                                     :try

                                     :if self.parent.is_camera_shared
                                     self.parent.is_camera_shared = False
        (self.share_camera_button.setIcon(self.share_camera_off_icon
                                     ()self.Network.close_camera_stream
        ()self.parent.update_share_camera_thread
                                     :else

                                     :()if check_active_cameras
                                     self.parent.is_camera_shared = True
        (self.share_camera_button.setIcon(self.share_camera_on_icon
                                     ()self.Network.start_camera_stream
        ()self.parent.start_camera_data_thread
                                     :else

        ("print("tried share camera but no camera is connected
                                     :except Exception as e
        ("{print(f"error in sharing or closing share camera error is: {e

        :(def share_screen_and_unshare(self
                                     :try

                                     :if self.parent.is_screen_shared
                                     self.parent.is_screen_shared = False
        (self.share_screen_button.setIcon(self.share_screen_off_icon

```

```

        ()self.Network.close_screen_stream
    ()self.parent.update_share_screen_thread
    :else
        self.parent.is_screen_shared = True
    (self.share_screen_button.setIcon(self.share_screen_on_icon
        ()self.parent.start_share_screen_send_thread
        ()self.Network.start_screen_stream
    :except Exception as e
        ("{print(f"error in sharing or closing share screen error is: {e

    : (def accept_call(self
        Add your logic when the call is accepted #
    (self.Network.send_accept_call_with(self.parent.getting_called_by

    : (def reject_call(self
        Add your logic when the call is rejected #
    (self.Network.send_reject_call_with(self.parent.getting_called_by
        ()self.parent.stop_sound
        ()self.parent.reset_call_var

    : (def ringing_user(self, name
        self.parent.is_calling = True
        self.parent.calling_to = name
        :try
        ()self.parent.updated_chat
        :except Exception as e
        ("{print(f"error ringing_user {e

```

```

        : (def call_user(self
            : try
if not self.parent.is_getting_called and not self.parent.is_calling and not
        : self.parent.is_in_a_call
            : ("") if self.parent.selected_chat.startswith
print(f"Calling Group...{self.parent.selected_chat}") # Replace this
        with your actual functionality
            : else
print(f"Calling User...{self.parent.selected_chat}") # Replace this
        with your actual functionality
            media_content =
QMediaContent(QUrl.fromLocalFile('discord_app_assets/Phone_Internal_Ringin
        ('gCalling - Sound Effect.mp3
            (self.parent.play_calling_sound_effect(media_content
            (self.Network.send_calling_user(self.parent.selected_chat
            (self.ringing_user(self.parent.selected_chat
            : except Exception as e
            (" {print(f"error call_user {e

: (def on_friend_button_clicked(self, label
            : try
            (self.selected_chat_changed(label
            : except Exception as e
            (" {print(f"error on_friend_button_clicked {e

: (def load_image_from_bytes_to_label(self, image_bytes, label
            () pixmap = QPixmap
            (pixmap.loadFromData(image_bytes

: if pixmap.width() == 0 or pixmap.height() == 0

```



```

        ("print("there is a error with image_bytes
                                return

Calculate the scaled size while maintaining the aspect ratio #
        ()aspect_ratio = pixmap.width() / pixmap.height
                                target_width = self.image_width
        (target_height = int(target_width / aspect_ratio
                                Scale the image to the target size #
        pixmap = pixmap.scaled(self.image_width, target_height,
                                (Qt.KeepAspectRatio
label.setGeometry(100, 100, self.image_width, target_height) # Adjust size
                                                                as needed

                                (label.setPixmap(pixmap

```

```

:(def load_image_from_bytes_to_button(self, image_bytes, button
                                (image = QImage.fromData(image_bytes

```

```

:if image.isNull() or image.width() == 0 or image.height() == 0
        ("print("There is an error with image_bytes
                                return

```

```

Calculate the scaled size while maintaining the aspect ratio #
        ()aspect_ratio = image.width() / image.height
                                target_width = self.image_width
        (target_height = int(target_width / aspect_ratio

                                Scale the image to the target size #
        scaled_image = image.scaled(target_width, target_height,
                                (Qt.KeepAspectRatio

```

```

Convert the QImage to QPixmap for displaying in the button #
    (pixmap = QPixmap.fromImage(scaled_image

                                Set the button's icon #
                                ((button.setIconSize(pixmap.size
                                ((button.setIcon(QIcon(pixmap
button.setGeometry(100, 100, self.image_width, target_height) # Adjust
                                size as needed

:(def show_context_menu(self, pos, button, file_bytes, type, file_name
                                (menu = QMenu(self
                                ""menu.setStyleSheet(f
                                }} QMenu
                                /* color: white; /* Text color of menu items
                                /* border: 1px solid gray; /* Border style
                                {{
                                }} QMenu::item:selected
background-color: {self.parent.standard_hover_color}; /* Hover color
                                /* when item is selected
                                {{
                                (""
                                ("download_action = menu.addAction("Download
                                download_action.triggered.connect(lambda:
                                ((download_file_from_bytes(file_bytes, type, file_name

Use the position of the button as the reference for menu placement #
                                (global_pos = button.mapToGlobal(pos

Show the context menu at the adjusted position #

```

```

(menu.exec_(global_pos

:(def create_temp_message_label(self, message
                                :try
                                    (label = QLabel(message, self
("label.setStyleSheet(f"color: white
(font = QFont(self.parent.font_text
(font.setPixelSize(self.parent.font_size
                                (label.setFont(font
number_of_rows = math.floor(len(message) / 160) + 1
                                :if len(message) > 0 and number_of_rows > 1
(format_label_text_by_row(label, message, number_of_rows
                                ()label.adjustSize
                                    return label
                                :except Exception as e
("){print(f"error in creating message label {e
                                    return None

:(def check_editing_status(self
()return self.text_entry.hasFocus

:(def garbage_button_clicked(self
self.parent.file_to_send = None
    "" = self.parent.file_name
()self.parent.update_chat_page_without_messages

:(def selected_chat_changed(self, name
    :if name != self.parent.selected_chat

```

```

        self.parent.is_new_chat_clicked = True
        :(")")if name.startswith
        self.parent.is_current_chat_a_group = True
        (text, _ = gets_group_attributes_from_format(name
        ("print("chat is group
        :else
        text = name
        self.parent.is_current_chat_a_group = False
        ("print("chat is a private chat
        ("{print(f"chat changed to {name
        (self.chat_name_label.setText(text
        place_holder_text = "Message" + " " + text
        :try
        :if self.text_entry
        (self.text_entry.setPlaceholderText(place_holder_text
        :except Exception as e
        ("{print(f"error selected_chat_changed {e
        self.parent.selected_chat = name
        self.parent.chat_start_index = None
        (self.Network.updated_current_chat(name
        ()self.image_too_big.hide
        self.parent.size_error_label = False
        self.parent.file_to_send = None
        self.parent.is_create_group_inside_chat_pressed = False
        "" = self.parent.file_name
        ()self.parent.updated_chat

        :(def is_mouse_on_chat_box(self, mouse_pos

```

```

()box_geometry = self.square_label.geometry
(return box_geometry.contains(mouse_pos

```

```

:(class MessageBox(QWidget
:(def __init__(self, parent, width, height, x, y
    ()__super().__init
    self.parent = parent
    self.width = width
    self.height = height
self.main_page_object = self.parent.parent
    self.x = x
    self.y = y
    ()self.init_ui

:(def init_ui(self
    Create a scroll area #
    :try
(self.scroll_area = QScrollArea(self.parent
    """)self.scroll_area.setStyleSheet
        } QScrollArea
        ;border: none
        {
        } QScrollBar:vertical
        ;border: none
        {
        } QScrollBar:horizontal
        ;border: none

```

```

        {
        ("""

        (self.scroll_area.setWidgetResizable(True

        Create a widget to contain labels and buttons #
        ()inner_widget = QWidget
        spacer = QSpacerItem(20, 40, QSizePolicy.Minimum,
        (QSizePolicy.Expanding
        (len_message_list = len(self.parent.parent.list_messages

        Set fixed width for inner widget to ensure proper layout #
        (inner_widget.setFixedWidth(380#

        Create a layout for the inner widget #
        (self.layout = QVBoxLayout(inner_widget
        self.space_between_widgets = 10
        self.layout.setSpacing(self.space_between_widgets) # Adjust this value
        as needed
        self.layout.setAlignment(Qt.AlignLeft | Qt.AlignTop) # Align widgets to the
        left and top

        Add labels and buttons to the layout #
        ((self.load_all_message_func(reversed(self.parent.parent.list_messages

        Set the inner widget as the scroll area's widget #
        (self.scroll_area.setWidget(inner_widget
        self.scroll_area.setGeometry(self.x, self.y, self.width, self.height) # Set
        the geometry directly
        :if self.parent.parent.is_new_chat_clicked
        ()maximum = self.scroll_area.verticalScrollBar().maximum

```

```

(self.scroll_area.verticalScrollBar()).setValue(maximum
    ("{print(f'Scrolled to maximum {maximum
        (self.scroll_value_changed(maximum
            Reset the flag #
        self.parent.parent.is_new_chat_clicked = False
            :else
        :if self.parent.parent.chat_start_index is not None

(self.scroll_area.verticalScrollBar()).setValue(self.parent.parent.chat_start_index

self.scroll_area.verticalScrollBar().valueChanged.connect(self.scroll_value_chan
    (ged
        :except Exception as e
    ("{print(f'Error in creating messages box {e

        :(def scroll_maximum(self
    ()maximum = self.scroll_area.verticalScrollBar().maximum
    (self.scroll_area.verticalScrollBar()).setValue(maximum

        :(def update_scroll_area_parent(self, new_parent
            (self.setParent(new_parent
            self.parent = new_parent
            (self.scroll_area.setParent(new_parent

        :(def load_all_message_func(self, message_list
            :for i in message_list
            (self.add_message_to_layout(i

        :(def add_message_to_layout(self, message

```

```

(self.add_or_insert_message_to_layout(message, False

:(def insert_message_to_layout(self, message
(self.add_or_insert_message_to_layout(message, True

:(def insert_messages_list_to_layout(self, message_list
:for i in message_list
(self.insert_message_to_layout(i

:(def add_new_message_at_start(self, message_dict
(self.add_message_to_layout(message_dict

:(def add_or_insert_message_to_layout(self, message, is_insert
["basic_files_types = ["xlsx", "py", "docx", "pptx", "txt", "pdf
i = message
("message_content = i.get("content
("message_time = i.get("timestamp
("message_sender = i.get("sender_id
("message_type = i.get("message_type
("file_name = i.get("file_name
:"if not message_content or message_type == "string
:if self.main_page_object.censor_data_from_strangers
if message_sender not in self.main_page_object.friends_list and
:message_sender != self.main_page_object.username
message_content =
(replace_non_space_with_star(message_content
content_label =
(self.parent.create_temp_message_label(message_content

```



```

                second part = Name + timestamp #
            ("")title_label = self.parent.create_temp_message_label
                )title_label.setText

f'<span style="font-size: {self.main_page_object.font_size + 2}px; color:
        '<white; font-weight: bold; ">{message_sender}</span
f'<span style="font-size: {self.main_page_object.font_size - 3}px; color:
        ('<gray; "> {message_time}</span
                :if not is_insert
                    (self.layout.addWidget(title_label
                    (self.layout.addWidget(content_label
                        :else
                    (self.layout.insertWidget(0, content_label
                    (self.layout.insertWidget(0, title_label
                        :elif message_type == "image
                            :try
                                decoded_compressed_image_bytes =
                                    (base64.b64decode(message_content
(image_bytes = zlib.decompress(decoded_compressed_image_bytes

                (image_label = QPushButton(self
image_label.setStyleSheet("background-color: transparent; border:
                                                                    (";none

                self.parent.load_image_from_bytes_to_button(image_bytes,
                                                                    (image_label
image_label.setMaximumWidth(int(self.width / 3)) # Adjust the
                maximum width as needed

image_label.clicked.connect(lambda _, image_bytes=image_bytes:
                            ((open_image_bytes(image_bytes
                :if self.main_page_object.censor_data_from_strangers

```

```

        if message_sender not in self.main_page_object.friends_list and
            :message_sender != self.main_page_object.username

image_label.setGraphicsEffect(QGraphicsBlurEffect(self.main_page_object.blur_
                                                    ((effect

        (image_label.setContextMenuPolicy(Qt.CustomContextMenu
            )image_label.customContextMenuRequested.connect
            lambda pos, file_bytes=image_bytes, button=image_label,
                ,type=message_type
, name=file_name: self.parent.show_context_menu(pos, button
            ((file_bytes, type, name

                "" = message

        (title_label = self.parent.create_temp_message_label(message
            )title_label.setText

f'<span style="font-size: {self.main_page_object.font_size + 2}px;
    '<color: white; font-weight: bold;">{message_sender}</span

f'<span style="font-size: {self.main_page_object.font_size - 3}px;
    ('<color: gray;"> {message_time}</span

                :if not is_insert

                (self.layout.addWidget(title_label
                (self.layout.addWidget(image_label
                    :else

                (self.layout.insertWidget(0, image_label
                (self.layout.insertWidget(0, title_label
                    :except Exception as e
                ("{'print(f'error in show messages is:{e
                    :elif message_type == "video
                        :try

```

```

        decoded_compressed_video_bytes =
            (base64.b64decode(message_content

(video_bytes = zlib.decompress(decoded_compressed_video_bytes

        (video_label = QPushButton(self

video_label.setStyleSheet("background-color: transparent; border:

                                                                    (";none

        (first_video_frame_bytes = extract_first_frame(video_bytes

        self.parent.load_image_from_bytes_to_button(first_video_frame_bytes,
                                                                    (video_label

        video_label.setMaximumWidth(int(self.width / 3)) # Adjust the
                                                                    maximum width as needed

        :if self.main_page_object.censor_data_from_strangers

        if message_sender not in self.main_page_object.friends_list and
            :message_sender != self.main_page_object.username

video_label.setGraphicsEffect(QGraphicsBlurEffect(self.main_page_object.blur_
                                                                    ((effect

                                                                    )video_label.clicked.connect

        lambda _, video_bytes=video_bytes:
            ((self.parent.parent.start_watching_video(video_bytes

        (play_button = QPushButton(video_label

"play_button_icon_path = "discord_app_assets/play_video_icon.png

        (play_button_size = (50, 50

        )play_button.clicked.connect

        lambda _, video_bytes=video_bytes:
            ((self.parent.parent.start_watching_video(video_bytes

        set_button_icon(play_button, play_button_icon_path,
            ([play_button_size[0], play_button_size[1

```

```

        (make_q_object_clear(play_button
        (layout = QHBoxLayout(video_label
layout.addWidget(play_button) # Add the play button to the layout

```

```

        Set the alignment and margins explicitly #
        (layout.setAlignment(Qt.AlignCenter
        (layout.setContentsMargins(0, 0, 0, 0
        layout.setSpacing(0) # Ensure no spacing between widgets

```

```

        Set the layout to the audio label #
        (video_label.setLayout(layout
        (video_label.setContextMenuPolicy(Qt.CustomContextMenu
        )video_label.customContextMenuRequested.connect
        lambda pos, file_bytes=video_bytes, button=video_label,
        ,type=message_type
        ,name=file_name: self.parent.show_context_menu(pos, button
        ((file_bytes, type, name

```

```

        "" = message
        (title_label = self.parent.create_temp_message_label(message
        )title_label.setText
        f'<span style="font-size: {self.main_page_object.font_size + 2}px;
        '<color: white; font-weight: bold;">{message_sender}</span
        f'<span style="font-size: {self.main_page_object.font_size - 3}px;
        ('<color: gray;"> {message_time}</span
        :if not is_insert
        (self.layout.addWidget(title_label
        (self.layout.addWidget(video_label
        :else

```

```

        (self.layout.addWidget(0, video_label
        (self.layout.addWidget(0, title_label
                                :except Exception as e
        ("{print(f"error in show messages is:{e
                                :elif message_type == "audio
                                :try
                                decoded_compressed_audio_bytes =
                                    (base64.b64decode(message_content
(audio_bytes = zlib.decompress(decoded_compressed_audio_bytes

        (audio_label = QPushButton(f"{file_name}", self
                                )audio_label.setStyleSheet
f"background-color: {self.main_page_object.standard_hover_color};
    border: none; color: white; font-size: {self.main_page_object.font_size}px;
    (";padding-left: 10%; margin: 0
        (audio_label.setFixedHeight(30
        (play_button = QPushButton(audio_label
"play_button_icon_path = "discord_app_assets/play_video_icon.png
        (play_button_size = (25, 25
        set_button_icon(play_button, play_button_icon_path,
            ([play_button_size[0], play_button_size[1
        )play_button.clicked.connect
        lambda _, audio_bytes=audio_bytes:
            ,play_mp3_from_bytes(audio_bytes

        ((self.parent.parent.mp3_message_media_player
            (layout = QHBoxLayout(audio_label
        layout.addWidget(play_button) # Add the play button to the layout

        Set the alignment and margins explicitly #

```

```

        (layout.setAlignment(Qt.AlignLeft
        (layout.setContentsMargins(0, 0, 0, 0
        layout.setSpacing(0) # Ensure no spacing between widgets

        Set the layout to the audio label #
        (audio_label.setLayout(layout
        (audio_label.setGeometry(x_pos, y, 300, 40 #
        (make_q_object_clear(play_button
        (audio_label.setContextMenuPolicy(Qt.CustomContextMenu
        )audio_label.customContextMenuRequested.connect
        lambda pos, file_bytes=audio_bytes, button=audio_label,
        ,type=message_type
        ,name=file_name: self.parent.show_context_menu(pos, button
        ((file_bytes, type, name

        "" = message

        (title_label = self.parent.create_temp_message_label(message
        )title_label.setText

        f'<span style="font-size: {self.main_page_object.font_size + 2}px;
        '<color: white; font-weight: bold;">{message_sender}</span

        f'<span style="font-size: {self.main_page_object.font_size - 3}px;
        ('<color: gray;"> {message_time}</span

        :if not is_insert

        (self.layout.addWidget(title_label
        (self.layout.addWidget(audio_label

        :else

        (self.layout.insertWidget(0, audio_label
        (self.layout.insertWidget(0, title_label

```

```

        :except Exception as e
        ("print("error in audio file
        :elif message_type in basic_files_types
        :try
        decoded_compressed_file_bytes =
            (base64.b64decode(message_content
        (file_bytes = zlib.decompress(decoded_compressed_file_bytes

        (link_label = QPushButton(f"{file_name}", self
        )link_label.setStyleSheet
        f"background-color: {self.main_page_object.standard_hover_color};
        border: none; color: white; font-size: {self.main_page_object.font_size}px;
        (";padding-left: 50%

        :if message_type == "txt
        link_label.clicked.connect(lambda _, file_bytes=file_bytes:
            ((open_text_file_from_bytes(file_bytes
        :elif message_type == "pptx
        )link_label.clicked.connect
        lambda _, file_bytes=file_bytes:
            ((open_pptx_from_bytes(file_bytes
        :elif message_type == "py
        )link_label.clicked.connect
        ((lambda _, file_bytes=file_bytes: open_py_from_bytes(file_bytes
        :elif message_type == "docx
        )link_label.clicked.connect
        lambda _, file_bytes=file_bytes:
            ((open_docx_from_bytes(file_bytes
        :elif message_type == "xlsx
        )link_label.clicked.connect
        lambda _, file_bytes=file_bytes:
            ((open_xlsx_from_bytes(file_bytes

```

```

        :elif message_type == "pdf
        )link_label.clicked.connect
((lambda _, file_bytes=file_bytes: open_pdf_from_bytes(file_bytes
        (link_label.setContextMenuPolicy(Qt.CustomContextMenu
        )link_label.customContextMenuRequested.connect
        lambda pos, file_bytes=file_bytes, button=link_label,
        ,type=message_type
,name=file_name: self.parent.show_context_menu(pos, button
        ((file_bytes, type, name
        (link_label.setFixedHeight(30

        (link_label.setGeometry(x_pos, y, 300, 40 #
        "" = message
        (title_label = self.parent.create_temp_message_label(message
        )title_label.setText
f'<span style="font-size: {self.main_page_object.font_size + 2}px;
    '<color: white; font-weight: bold;">{message_sender}</span
f'<span style="font-size: {self.main_page_object.font_size - 3}px;
    ('<color: gray;"> {message_time}</span

        :if not is_insert
        (self.layout.addWidget(title_label
        (self.layout.addWidget(link_label
        :else
        (self.layout.insertWidget(0, link_label
        (self.layout.insertWidget(0, title_label
        :except Exception as e
        ("{print(f"error in show messages is:{e

```



```

                                : (def clear_layout(self
                                : () while self.layout.count
                                (item = self.layout.takeAt(0
                                () widget = item.widget
                                : if widget
                                () widget.deleteLater

                                : (def update_messages_layout(self
                                    () self.initUI

                                : (def scroll_to_index(self, index
                                    Get the vertical scroll bar of the scroll area #
                                    () scroll_bar = self.scroll_area.verticalScrollBar

                                    Set the scroll bar value to scroll to the specified index #
                                    (scroll_bar.setValue(index

                                : (def scroll_up_by_n_widgets(self, n
                                    total_height = 0
                                : (((for i in range(min(n * 2, self.layout.count
                                    () widget = self.layout.itemAt(i).widget
                                    : if widget
                                    total_height += widget.sizeHint().height() +
                                        self.space_between_widgets
                                    ("{print(f"total height is {total_height
                                    (self.scroll_area.verticalScrollBar().setValue(total_height
                                    ((self.scroll_value_changed(self.scroll_area.verticalScrollBar().value

                                : (def scroll_value_changed(self, value

```

I don't want to ask for more message when I just load the messages in #

```
:if value == 0 and not self.parent.parent.is_new_chat_clicked
    :if len(self.main_page_object.list_messages) >= 15
        ()self.main_page_object.Network.ask_for_more_messages
            ("print("asked for more messages
```

```
self.main_page_object.chat_start_index = value
```

```
:(class CreateGroupBox(QWidget
:
:(def __init__(self, parent, x, y, box_format
    ()__super().__init
    self.parent = parent
    :try
        self.x = x
        self.y = y
        self.box_format = box_format
    self.create_group_open_x = self.parent.create_group_open_x
    self.create_group_open_y = self.parent.create_group_open_y
        self.selected_group_members =
        self.parent.parent.selected_group_members
self.group_max_members = self.parent.parent.group_max_members
    self.friends_list = self.parent.parent.friends_list
self.standard_hover_color = self.parent.parent.standard_hover_color
    self.create_group_index = self.parent.parent.create_group_index
        :except Exception as e
            ("print(f"error in initiating create group box {e
                ()self.init_ui
```

```

                                :(def init_ui(self
                                :try
                                :if self.box_format == "create
                                "submit_button_text = "Create DM
                                :else
                                "submit_button_text = "ADD
                                starter_x = self.x
                                starter_y_of_border = self.y + 50
                                adding_border_height = 400
                                adding_border_width = 300

                                (border_of_adding = QLabel(self.parent
                                border_of_adding.setGeometry(starter_x, starter_y_of_border,
                                (adding_border_width, adding_border_height
                                ()_border_of_adding.raise
                                ;border_of_adding.setStyleSheet("""border: 2px solid black
                                /* Use a slightly darker shade for the border */
                                (""";border-radius: 5px

                                (label = QLabel(f"Select friends", self.parent
                                (""";label.setStyleSheet("""color: white;font-size: 20px
                                (label.move(starter_x + 20, starter_y_of_border + 10
                                Page = 0
                                :if len(self.friends_list) > 0
                                Page = self.create_group_index + 1

                                :if self.parent.parent.is_create_group_inside_chat_pressed
                                :if self.parent.current_group_id

```

```

        number_of_group_members =
        )self.parent.parent.get_number_of_members_by_group_id
        (self.parent.current_group_id
        page_plus_selected_label_text = f"You can add
        {self.group_max_members - number_of_group_members -
        "len(self.selected_group_members)} more friends

        page_plus_selected_text =
        f"Page({Page})/{calculate_division_value(len(self.friends_list))}
        "({(Selected({len(self.selected_group_members
        :else

        page_plus_selected_label_text = f"You can add
        {(self.group_max_members - 2) - len(self.selected_group_members)} more
        "friends

        page_plus_selected_text =
        f"Page({Page})/{calculate_division_value(len(self.friends_list))}
        "({(Selected({len(self.selected_group_members
        :else

        page_plus_selected_label_text = f"You can add
        {(self.group_max_members - 1) - len(self.selected_group_members)} more
        "friends

        page_plus_selected_text =
        f"Page({Page})/{calculate_division_value(len(self.friends_list))}
        "({(Selected({len(self.selected_group_members
self.page_plus_selected_label = QLabel(page_plus_selected_label_text,
                                     (self.parent
self.page_plus_selected_label.setStyleSheet("""color: white;font-size:
                                     (""";14px
self.page_plus_selected_label.move(starter_x + 20, starter_y_of_border
                                     (+ 45

        self.amount_of_people_to_add_text_label =
        (QLabel(page_plus_selected_text, self.parent
        self.amount_of_people_to_add_text_label.setStyleSheet("""color:
        (""";white;font-size: 12px

```

```

self.amount_of_people_to_add_text_label.move(starter_x + 40,
                                              (starter_y_of_border + 75

                                              ""style_sheet = f
                                              }} QPushButton
                                              ;color: white
                                              ;font-size: 16px

/* background-color: rgba(0, 0, 0, 0); /* Transparent background
border: 2px solid {self.standard_hover_color}; /* Use a slightly darker
                                              /* shade for the border

                                              ;border-radius: 5px
                                              {{
                                              }} QPushButton: hover
                                              ;background-color: #2980b9
                                              {{
                                              ""

(scroll_up_button = QPushButton("↑", self.parent
(scroll_up_button.move(starter_x + 230, starter_y_of_border + 25
                      (scroll_up_button.setFixedWidth(50
                      (scroll_up_button.setStyleSheet(style_sheet

scroll_up_button.clicked.connect(partial(self.parent.handle_create_group_index,
                                      (""up

(scroll_down_button = QPushButton("↓", self.parent
(scroll_down_button.move(starter_x + 230, starter_y_of_border + 55
                      (scroll_down_button.setFixedWidth(50
                      (scroll_down_button.setStyleSheet(style_sheet

```

```

scroll_down_button.clicked.connect(partial(self.parent.handle_create_group_inde
                                     ("x", "down

                                     starter_x = self.x
                                     starter_y = self.y + 150
                                     i = 0
                                     :for friend in self.friends_list
:if self.create_group_index * 5 <= i < (self.create_group_index + 1) * 5
    (friend_label = QPushButton(friend, self.parent
    friend_label.friend_name = friend
    ""friend_label.setStyleSheet(f
        }} QPushButton
        ;color: white
        ;font-size: 18px
    ;{border: 2px solid {self.standard_hover_color
        ;border-radius: 5px
        ;padding: 5px
        ;margin-bottom: 18px
    /* text-align: left; /* Align text to the left
        {{

        }} QPushButton:hover
    /* background-color: #3498db; /* Bluish hover color
        {{
        ("
        (friend_checkbox = QCheckBox(self.parent
:if self.parent.parent.is_create_group_inside_chat_pressed
    :if self.parent.current_group_id

```

```

        group_members =
        )self.parent.parent.get_group_members_by_group_id
        (self.parent.current_group_id
        :if friend in group_members
        (friend_checkbox.setChecked(True
        :else

        ((friend_label.clicked.connect(partial(self.parent.toggle_checkbox, self
        :else

        :if self.parent.parent.selected_chat == friend
        (friend_checkbox.setChecked(True
        :else

        ((friend_label.clicked.connect(partial(self.parent.toggle_checkbox, self
        :else

friend_label.clicked.connect(partial(self.parent.toggle_checkbox,
                                                                    ((self
                                                                    :if friend in self.selected_group_members
                                                                    (friend_checkbox.setChecked(True
friend_checkbox.friend_name = friend # Store friend's name as an
                                                                    attribute

        (friend_checkbox.stateChanged.connect(self.parent.friend_checkbox_changed
            height = friend_label.height() + 30
            friend_label.setGeometry(starter_x + 10, starter_y,
                                    (adding_border_width - 20, height
        (friend_checkbox.move(starter_x + 260, starter_y + 15
            starter_y += friend_label.height() - 20
            ()_friend_label.raise
            ()_friend_checkbox.raise
            i += 1

```

```
(button = QPushButton(submit_button_text, self.parent
button.move(starter_x + 15, starter_y_of_border + adding_border_height
```

(- 80

```
(button.setFixedHeight(self.parent.friends_button_height
                        : "if self.box_format == "create
(button.clicked.connect(self.parent.create_dm_pressed
                        : else
(button.clicked.connect(self.parent.add_users_to_group
```

```
        """button.setStyleSheet(f
                                }} QPushButton
;{background-color: {self.parent.parent.background_color_hex
;{border: 2px solid {self.parent.parent.standard_hover_color
                                ;border-radius: 5px
                                ;padding: 8px 16px
                                ;color: #b9c0c7
                                ;font-family: Arial, sans-serif
                                ;font-size: 14px
                                ;font-weight: normal
                                ;(box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1
                                {{
                                }} QPushButton: hover
                                ;background-color: #2980b9
                                {{
                                }} QPushButton: pressed
                                ;background-color: #202225
```



```

;border-color: #72767d
    {{
    (""

(button.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Fixed
    (button.setFixedWidth(adding_border_width - 30
        :except Exception as e
        ("{print(f"error in creating group box {e

        :(def update_labels_text(self
            Page = 0
            :if len(self.friends_list) > 0
            Page = self.create_group_index + 1
        :if self.parent.parent.is_create_group_inside_chat_pressed
            :if self.parent.current_group_id
                number_of_group_members =
            )self.parent.parent.get_number_of_members_by_group_id
                (self.parent.current_group_id
            page_plus_selected_label_text = f"You can add
            {self.group_max_members - number_of_group_members -
            "len(self.selected_group_members)} more friends
            page_plus_selected_text =
            f"Page({Page})/{calculate_division_value(len(self.friends_list))}
            "({(Selected({len(self.selected_group_members
            :else
            page_plus_selected_label_text = f"You can add
            {(self.group_max_members - 2) - len(self.selected_group_members)} more
            "friends
            page_plus_selected_text =
            f"Page({Page})/{calculate_division_value(len(self.friends_list))}
            "({(Selected({len(self.selected_group_members

```

```

:else

        page_plus_selected_label_text = f"You can add
        {(self.group_max_members - 1) - len(self.selected_group_members)} more
        "friends

        page_plus_selected_text =
        f"Page({Page})/{calculate_division_value(len(self.friends_list))}
        "({(Selected({len(self.selected_group_members
        (self.page_plus_selected_label.setText(page_plus_selected_label_text
        (self.amount_of_people_to_add_text_label.setText(page_plus_selected_text

class FriendsChatListWidget(QWidget
:
    def __init__(self, chat_box_object, chats_list
        super().__init__
        self.chat_box_object = chat_box_object
        self.friends_button_height = 50
        self.draw_friends_buttons(chats_list

    def draw_friends_buttons(self, friend_list
friend_starter_y = 170 + (self.chat_box_object.parent.chat_box_chats_index
(* -50
        friend_x = 250
        if friend_list is not None
        for friend in friend_list
        try
            button = self.chat_box_object.create_friend_button(friend, (friend_x,
                                                                    ((friend_starter_y
                                                                    button.setGeometry(friend_x, friend_starter_y, 100,
                                                                    (self.chat_box_object.friends_button_height
            friend_starter_y += self.chat_box_object.friends_button_height
        except Exception as e

```

```
("{print(f"Error in drawing friends button: {e
```

```
import socket
import json
import zlib
import threading
import logging
import base64
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives import padding as aes_padding
import secrets
import pickle
'vc_data_sequence = br'\vc_data
'share_screen_sequence = br'\share_screen_data
'share_camera_sequence = br'\share_camera_data

:(def slice_up_data(data, mtu
    [] = sliced_data
:(for start_index in range(0, len(data), mtu
    end_index = start_index + mtu
([sliced_data.append(data[start_index:end_index
    return sliced_data
```

```

:(def create_dictionary_with_message_type(message_type, keys, values
    Ensure both lists have the same length #
    :(if len(keys) != len(values
        ("raise ValueError("Lists must have the same length

    Add the message_type as the first key-value pair #
    ('keys.insert(0, 'message_type
    (values.insert(0, message_type

    Create the dictionary using a dictionary comprehension #
    {((result = {keys[i]: values[i] for i in range(len(keys

    return result

:())def generate_secure_symmetric_key
    (symmetric_key = secrets.token_bytes(32
    return symmetric_key

:())def generate_aes_key
    Generate a random 256-bit (32-byte) key for AES-256 #
    (aes_key = secrets.token_bytes(32
    return aes_key

```

```

        :()def generate_rsa_key_pair
    )private_key = rsa.generate_private_key
        ,public_exponent=65537
        ,key_size=2048
        ()backend=default_backend
        (
return private_key.public_key(), private_key

:()def encrypt_with_rsa(public_key, data
    )ciphertext = public_key.encrypt
        ,data
        )padding.OAEP
,((()mgf=padding.MGF1(algorithm=hashes.SHA256
    ,()algorithm=hashes.SHA256
        label=None
        (
        (
("return base64.b64encode(ciphertext).decode("utf-8

:()def decrypt_with_rsa(private_key, ciphertext
    (ciphertext = base64.b64decode(ciphertext
        )plaintext = private_key.decrypt
            ,ciphertext
            )padding.OAEP
,((()mgf=padding.MGF1(algorithm=hashes.SHA256
    ,()algorithm=hashes.SHA256

```



```

        Use PKCS#7 unpadding #
        ()unpadder = aes_padding.PKCS7(128).unpadder
    ()unpadded_data = unpadder.update(decrypted_data) + unpadder.finalize
        return type bytes #
        return unpadded_data
    :except Exception as e
    ({print(f"Error in decryption: {e
        return 1

:(def send_data_in_chunks(sock, data
    .Send data over a socket in chunks""

    :Args

    .sock (socket.socket): The socket object for sending data
    .data (bytes): The data to be sent

    :Returns

    .bool: True if the data was sent successfully, False otherwise
        ""

    :try

        Define the chunk size #
        chunk_size = 4096 # Adjust this based on your requirements

        Send data in chunks #
        bytes_sent = 0
        :(while bytes_sent < len(data
            [chunk = data[bytes_sent:bytes_sent + chunk_size

```

```

        (bytes_sent += sock.send(chunk

                                return True

                                :except Exception as e
                                ({print(f"Error sending data: {e
                                return False

                                :class ClientNet
                                :(def __init__(self
                                (__self.logger = logging.getLogger(__name
                                (self.logger.setLevel(logging.DEBUG

                                Create a StreamHandler with the desired format #
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s -
                                ('%(message)s
                                ()stream_handler = logging.StreamHandler
                                (stream_handler.setFormatter(formatter

                                Add the StreamHandler to the logger #
                                (self.logger.addHandler(stream_handler
self.client_tcp_socket = socket.socket(socket.AF_INET,
                                (socket.SOCK_STREAM
self.client_udp_socket = socket.socket(socket.AF_INET,
                                (socket.SOCK_DGRAM

                                "self.server_ip = "127.0.0.1
                                self.port = 5555
                                (self.addr = (self.server_ip, self.port

```



```

({self.logger.debug(f"trying to connect to addr: {self.addr
                                self.size = 0000000
                                self.original_len = 10
                                self.mtu = None
                                self.aes_key = None
                                self.connected = False
                                ()self.sending_tcp_data_lock = threading.Lock

                                :(def if_connected(self
                                ()self.connect_udp
                                ()self.initiate_rsa_protocol
                                ()self.check_max_packet_size_udp

                                :(def connect_tcp(self
                                :try
                                (self.client_tcp_socket.connect(self.addr
                                ("self.logger.info("tcp socket connected to server
                                ()self.if_connected
                                return True
                                :except Exception as e
                                ("{self.logger.info(f"couldn't connect tcp socket to server {e
                                return False

                                :(def connect_udp(self
                                :try
                                (self.client_udp_socket.connect(self.addr
                                ("self.logger.info("udp socket connected to server
                                ()address = self.client_udp_socket.getsockname

```

```

        (print("Socket address:", address
                :except
("self.logger.info("couldn't connect udp socket to server
                pass

        :(def send_bytes(self, data
                :try

Convert the length of the data to a string #
        ()self.sending_tcp_data_lock.acquire
                :if self.aes_key is None
        ((size_str = str(len(data
        ((size = str(self.size + int(size_str
(number_of_zero = self.original_len - len(size
        size = ("0" * number_of_zero) + size
                Send the size as a string #
        (('self.client_tcp_socket.send(size.encode('utf-8
                Send the actual data as bytes #
        (self.client_tcp_socket.send(data
                :else

(encrypted_data = encrypt_with_aes(self.aes_key, data
        ((size_str = str(len(encrypted_data
        ((size = str(self.size + int(size_str
(number_of_zero = self.original_len - len(size
        size = ("0" * number_of_zero) + size
                Send the size as a string #
        (('self.client_tcp_socket.send(size.encode('utf-8
                Send the actual data as bytes #
        (self.client_tcp_socket.send(encrypted_data

```

```

                                :except socket.error as e
                                    (print(e
                                        :finally
                                            Release the lock #
                                        ())self.sending_tcp_data_lock.release

:(def send_large_udp_data(self, data, data_type, shape_of_frame=None
                                :if len(data) > self.mtu
                                    ((sliced_data = slice_up_data(data, int(self.mtu * 0.8
                                        :else
                                            [sliced_data = [data

:(for i, data_slice in enumerate(sliced_data
                                } = message
                                    ,message_type": data_type"
                                        ,is_first": i == 0"
                                            ,is_last": i == len(sliced_data) - 1"
                                                ,sliced_data": data_slice"
                                                    shape_of_frame": shape_of_frame"
                                                        {
                                                            (self.send_message_dict_udp(message

:(def send_bytes_udp(self, data
                                :try
                                    Encrypt the data if encryption is enabled #
                                        :if self.aes_key is not None
                                            (encrypted_data = encrypt_with_aes(self.aes_key, data
                                                data = encrypted_data

```

```

        (self.client_udp_socket.sendto(data, self.addr
                                     :except socket.error as e
("{{print(f"error in sending udp {e} , data size = {len(data
                                     raise

```

```

:(def check_max_packet_size_udp(self
    'data = b'a
    :try
    :while True
        (self.send_bytes_udp(data
            (data += (b'a' * 100
            :except socket.error as e
("{{self.logger.info(f"Network mtu is: {len(data) - 10
        self.mtu = len(data) - 10

```

```

:(def send_message_dict_udp(self, message_dict
    :try
    (pickled_data = pickle.dumps(message_dict
        (self.send_bytes_udp(pickled_data
        :except Exception as e
        (print(e

```

```

:(def send_message_dict_tcp(self, message_dict
    :try
    (pickled_data = pickle.dumps(message_dict
        (self.send_bytes(pickled_data
        :except Exception as e

```

```

(print(e

:(def connect_between_udp_port_address_to_username(self
    ()udp_address = self.client_udp_socket.getsockname
    ()tcp_address = self.client_tcp_socket.getsockname
message = {"message_type": "connect_udp_port", "udp_address":
    {udp_address, "tcp_address": tcp_address
    (self.send_message_dict_tcp(message

:(def send_song_search(self, search_str
    :try
{message = {"message_type": "song_search", "search_str": search_str
    (self.send_message_dict_tcp(message
    :except socket.error as e
    (print(e

:(def send_remove_song_from_playlist(self, song_title
    :try
{message = {"message_type": "remove_song", "song_title": song_title
    (self.send_message_dict_tcp(message
    :except socket.error as e
    (print(e

:(def save_song_in_playlist(self, song_dict
    :try
{message = {"message_type": "save_song", "song_dict": song_dict
    (self.send_message_dict_tcp(message
    :except socket.error as e
    (print(e

```

```

:(def ask_for_song_bytes_by_playlist_index(self, index
                                          :try
message = {"message_type": "playlist_song_bytes_by_index", "index":
                                          {index
      (self.send_message_dict_tcp(message
                                   :except socket.error as e
                                   (print(e

:(def updated_current_chat(self, current_chat
                           :try
message = {"message_type": "current_chat", "current_chat":
                           {current_chat
      (self.send_message_dict_tcp(message
                                   :except socket.error as e
                                   (print(e

:(def ask_for_more_messages(self
                           :try
{"message = {"message_type": "more_messages
      (self.send_message_dict_tcp(message
                                   :except socket.error as e
                                   (print(e

:(def start_screen_stream(self
                           :try
message = {"message_type": "call", "call_action_type": "stream",
          {"stream_type": "ScreenStream", "action": "start
      (self.send_message_dict_tcp(message

```

```

                                :except socket.error as e
                                (print(e

                                :(def close_screen_stream(self
                                :try
message = {"message_type": "call", "call_action_type": "stream",
          {"stream_type": "ScreenStream", "action": "close
          (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

                                :(def start_camera_stream(self
                                :try

                                "stream_type = "CameraStream
message = {"message_type": "call", "call_action_type": "stream",
          {"stream_type": stream_type, "action": "start
          (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

                                :(def close_camera_stream(self
                                :try

                                "stream_type = "CameraStream
message = {"message_type": "call", "call_action_type": "stream",
          {"stream_type": stream_type, "action": "close
          (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def watch_screen_stream_of_user(self, user
                                :try
                                "stream_type = "ScreenStream
message = {"message_type": "call", "call_action_type": "stream",
          , "stream_type": stream_type
          {action": "watch", "user_to_watch": user"
          (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def watch_camera_stream_of_user(self, user
                                :try
                                "stream_type = "CameraStream
message = {"message_type": "call", "call_action_type": "stream",
          , "stream_type": stream_type
          {action": "watch", "user_to_watch": user"
          (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def stop_watching_current_stream(self
                                :try
                                , "message = {"message_type": "call", "call_action_type": "stream
                                {"action": "stop_watch"
                                (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def leave_call(self

```



```

:try

    Convert the length of the data to a string #
    , "message = {"message_type": "call", "call_action_type": "in_call_action"
                    {"action": "ended"
    (self.send_message_dict_tcp(message
                    :except socket.error as e
                    (print(e

:(def send_new_password(self, new_password
:try
    , "message = {"message_type": "password"
    {"action": "new_password", "new_password": new_password"
    (self.send_message_dict_tcp(message
                    :except socket.error as e
                    (print(e

:(def send_new_group_image_to_server(self, image_bytes, group_id
('encoded_b64_image = base64.b64encode(image_bytes).decode('utf-8
    , "message = {"message_type": "group
    action": "update_image", "group_id": group_id,"
    {"encoded_b64_image": encoded_b64_image
    (self.send_message_dict_tcp(message

:(def create_group(self, group_members_list
(json_group_members_list = json.dumps(group_members_list
    , "message = {"message_type": "group
{"action": "create", "group_members_list": json_group_members_list"
    (self.send_message_dict_tcp(message

```

```

:(def add_user_to_group(self, group_id, users_list
  , "message = {"message_type": "group", "action": "add_user
  {(group_id": group_id, "users_to_add": json.dumps(users_list"
    (self.send_message_dict_tcp(message

:(def send_calling_user(self, user_that_is_called
  :try
  , "message = {"message_type": "call", "call_action_type": "in_call_action
    {action": "calling", "calling_to": user_that_is_called"
    (self.send_message_dict_tcp(message
      :except socket.error as e
      (print(e

:(def stop_ringing_to_group_or_user(self
  :try
  message = {"message_type": "call", "call_action_type":
    , ""change_calling_status
    {"!action": "stop"
    (self.send_message_dict_tcp(message
      :except socket.error as e
      (print(e

:(def send_join_call_of_group_id(self, group_id
  :try
  , "message = {"message_type": "call", "call_action_type": "in_call_action
    {action": "join_call", "group_id": group_id"
    (self.send_message_dict_tcp(message
      :except socket.error as e
      (print(e

```

```

:(def send_accept_call_with(self, accepted_caller
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
              {action": "accepted_call", "accepted_caller": accepted_caller"
              (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def send_reject_call_with(self, rejected_caller
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
              {action": "rejected_call", "rejected_caller": rejected_caller"
              (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def toggle_mute_for_myself(self
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action
              {"action": "mute_myself"
              (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

```

```

:(def toggle_deafen_for_myself(self
                                :try
, "message = {"message_type": "call", "call_action_type": "in_call_action

```

```

        {"action": "deafen_myself"
    (self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_login_info(self, username, password

,message = {"message_type": "login", "username": username
            {password": password"
    (self.send_message_dict_tcp(message

:(def send_sign_up_info(self, username, password, email
        "message_format = "sign_up
,message = {"message_type": message_format, "username": username
            {password": password, "email": email"
    (self.send_message_dict_tcp(message

:(def send_security_token(self, security_token
        "message_format = "security_token
message = {"message_type": message_format, "security_token":
            {security_token
    (self.send_message_dict_tcp(message

def send_username_and_email_froget_password(self, username, password,
                                           :(email
        "message_format = "forget password
,message = {"message_type": message_format, "username": username
            {email": email"
    (self.send_message_dict_tcp(message

```

```

:(def send_message(self, sender, receiver, content, type, file_name
                        :(if isinstance(content, bytes
                                If content is bytes, encode it as a Base64 string #
                                ('content = base64.b64encode(content).decode('utf-8
                                "message_format = "add_message
, message = {"message_type": message_format, "sender": sender
                        , "receiver": receiver"
                        , "content": content"
                        , "type": type"
                        file_name": file_name"
                        {
                        (self.send_message_dict_tcp(message

:(def send_profile_pic(self, profile_pic
                        :(if isinstance(profile_pic, bytes
                                If content is bytes, encode it as a Base64 string #
                                ('content = base64.b64encode(profile_pic).decode('utf-8
                                :elif profile_pic is None
                                "str_profile_pic = "None
                                content = str_profile_pic
                                "message_format = "update_profile_pic
message = {"message_type": message_format, "b64_encoded_profile_pic":
                                                content
                                                {
                                                (self.send_message_dict_tcp(message

:(def send_vc_data(self, vc_data
                        :try

```

```

        full_message = vc_data
        (compressed_message = zlib.compress(full_message
            "message_format = "vc_data
        (self.send_large_udp_data(compressed_message, message_format
            :except Exception as e
            ("{print(f"error in send vc data is: {e

:(def send_share_screen_data(self, share_screen_data, shape_of_frame
    :try
        full_message = share_screen_data
        (compressed_message = zlib.compress(full_message
            "message_format = "share_screen_data
        self.send_large_udp_data(compressed_message, message_format,
            (shape_of_frame
            :except Exception as e
            ("{print(f"error is in send share screen data: {e

:(def send_share_camera_data(self, share_camera_data, shape_of_frame
    :try
        full_message = share_camera_data
        (compressed_message = zlib.compress(full_message
            "message_format = "share_camera_data
        self.send_large_udp_data(compressed_message, message_format,
            (shape_of_frame
            :except Exception as e
            ("{print(f"error is send share camera data: {e

:(def send_settings_dict_to_server(self, settings_dict
    :try

```

```

message = {"message_type": "settings_dict", "settings_dict": settings_dict
            {
                (self.send_message_dict_tcp(message
                    :except socket.error as e
                        (print(e

:(def send_friend_request(self, friend_username
                                :try
                                    Convert the length of the data to a string #
message = {"message_type": "friend_request", "username_for_request":
                                                friend_username
                                                    {
                (self.send_message_dict_tcp(message
                    :except socket.error as e
                        (print(e

:(def send_remove_friend(self, friend_username
                                :try
message = {"message_type": "friend_remove", "username_to_remove":
                                                friend_username
                                                    {
                (self.send_message_dict_tcp(message
                    :except socket.error as e
                        (print(e

:(def send_exit_group(self, group_id
                                :try
message = {"message_type": "exit_group", "group_to_exit_id": group_id
            {

```

```

        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def send_remove_chat(self, chat
                                     :try
message = {"message_type": "remove_chat", "chat_to_remove": chat
        {
        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def send_remove_user_from_group(self, user, group_id
                                     :try
message = {"message_type": "remove_user_from_group",
          "user_to_remove": user
          group_id": group_id" ,
          {
        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

    : (def block_user(self, user_to_block
                                     :try
message = {"message_type": "block", "user_to_block": user_to_block
        {
        (self.send_message_dict_tcp(message
                                     :except socket.error as e
                                     (print(e

```



```

:(def unblock_user(self, user_to_unblock
                                :try
message = {"message_type": "unblock", "user_to_unblock":
                                user_to_unblock
                                {
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_friends_request_rejection(self, rejected_user
                                :try
message = {"message_type": "friend_request_status", "action": "reject",
                                "rejected_user": rejected_user
                                {
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def send_friends_request_acception(self, accepted_user
                                :try
message = {"message_type": "friend_request_status", "action": "accept",
                                "accepted_user": accepted_user
                                {
(self.send_message_dict_tcp(message
                                :except socket.error as e
                                (print(e

:(def ask_for_security_token(self

```

```

:try
"message = {"message_type": "security_token", "action": "needed"
{
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_sign_up_verification_code(self, code
:try
message = {"message_type": "sign_up", "action": "verification_code",
"code": code
{
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_login_2fa_code(self, code
:try
message = {"message_type": "login", "action": "2fa", "code": code
{
(self.send_message_dict_tcp(message
:except socket.error as e
(print(e

:(def send_logout_message(self
:try
"message = {"message_type": "logout"
{
(self.send_message_dict_tcp(message

```

```

        except socket.error as e
            (print(e

    (def receive_by_size(self, size, buffer_size=16384
        ()received_data = bytearray

        :while len(received_data) < size
            (remaining_size = size - len(received_data
                :try
                    ((chunk = self.client_tcp_socket.recv(min(buffer_size, remaining_size
                        :except socket.error as e
                            Handle socket errors, e.g., connection reset #
                                ({print(f"Socket error: {e
                                    return None

                                :if not chunk
                                    Connection closed #
                                        return None

                                (received_data.extend(chunk

                                (return bytes(received_data

    (def recv_str(self
        :try
            Receive the size as binary data and convert it to an integer #
            ('size_str = self.client_tcp_socket.recv(self.original_len).decode('utf-8

```

Convert the size string to an integer #

(size = int(size_str

Receive the actual data based on the size #

(data = self.receive_by_size(size

:try

encrypted_data = data

:if data is None

("print("Received data is None

return None

(data = decrypt_with_aes(self.aes_key, encrypted_data

(return pickle.loads(data

:except Exception as e

("{print(f"error in receiving data: {e

return data

:except (socket.error, ValueError) as e

("{print(f"error in recv_str:{e

return None

:(def recv_bytes(self

:try

Receive the size as binary data and convert it to an integer #

('size_str = self.client_tcp_socket.recv(self.original_len).decode('utf-8

Convert the size string to an integer #

(size = int(size_str

```

        Receive the actual data based on the size #
        (data = self.receive_by_size(size
            return data

    except (socket.error, ValueError) as e
        ({print(f"Error: {e
return None # Return None in case of an error

    :(def recv_udp(self
(fragment_data, address = self.client_udp_socket.recvfrom(100000
return decrypt_with_aes(self.aes_key, fragment_data), address

    :(def return_socket(self
return self.client_tcp_socket

    :(def close(self
        :try
            ()self.client_tcp_socket.close
            ()self.client_udp_socket.close
        except socket.error as e
            (print(e

    :(def initiate_rsa_protocol(self
        create 256 bytes key #
        ()client_symmetric_key = generate_secure_symmetric_key
        'public_key_byte_sequence = br"server:public-key

the client receives the server Rsa public key #

```

```

        received_serialized_server_public_key_bytes = self.recv_bytes
        if
received_serialized_server_public_key_bytes.startswith(public_key_byte_sequence)
        : (ce
            received_serialized_server_public_key_bytes =
        [:(received_serialized_server_public_key_bytes[len(public_key_byte_sequence
            : else
                ("print("did not expect message
            return

```

```

        Deserialize the received public key #
    )server_public_key = serialization.load_pem_public_key
        ,received_serialized_server_public_key_bytes
        ()backend=default_backend
        (
encrypted_symmetric_key = encrypt_with_rsa(server_public_key,
        (client_symmetric_key

```

```

        Use send_bytes to send the encrypted key as bytes #
'symmetric_key_byte_sequence = br'\server:symmetric-key
        self.send_bytes(symmetric_key_byte_sequence +
            ("encrypted_symmetric_key.encode("utf-8

```

```

        ()encrypt_aes_key = self.recv_bytes
        : (if encrypt_aes_key.startswith(symmetric_key_byte_sequence
            encrypt_aes_key =
        [:(encrypt_aes_key[len(symmetric_key_byte_sequence
(aes_key = decrypt_with_aes(client_symmetric_key, encrypt_aes_key
        self.aes_key = aes_key

```

```
self.logger.info(f'Started to communicate with the server , with AES key  
                ({{self.aes_key
```

