# ***עבודת גמר בתכנון ותכנות מערכות התמחות סייבר***

# 5 יח"ל – שאלון 883589

# מערכת Cloud Walkie-Talkie

# מגיש: אריאל אורן (ת.ז: 215358904)

# מורה מנחה: שרית שוורץ

# בית הספר: אמי"ת גוש-דן בר-אילן

# שנת הגשה: 2023, תשפ"ג

תוכן עניינים

[***עבודת גמר בתכנון ותכנות מערכות התמחות סייבר*** 1](#_Toc137583835)

[5 יח"ל – שאלון 883589 1](#_Toc137583836)

[מערכת Cloud Walkie-Talkie 1](#_Toc137583837)

[מגיש: אריאל אורן (ת.ז: 215358904) 1](#_Toc137583838)

[מורה מנחה: שרית שוורץ 1](#_Toc137583839)

[בית הספר: אמי"ת גוש-דן בר-אילן 1](#_Toc137583840)

[שנת הגשה: 2023, תשפ"ג 1](#_Toc137583841)

[מבוא 3](#_Toc137583842)

[ייזום 3](#_Toc137583843)

[פירוט תיאור המערכת (אפיון) 3](#_Toc137583844)

[תיאור תחום הידע 4](#_Toc137583845)

[פירוט מעמיק של היכולות שהוצגו בשלב הקודם 4](#_Toc137583846)

[מבנה / ארכיטקטורה של הפרויקט 5](#_Toc137583847)

[Sign in 6](#_Toc137583848)

[מימוש הפרויקט 12](#_Toc137583849)

[מדריך למשתמש 17](#_Toc137583850)

[סיכום אישי / רפלקציה 21](#_Toc137583851)

[ביבליוגרפיה 22](#_Toc137583852)

[קוד 22](#_Toc137583853)

[Server.py 22](#_Toc137583854)

[client\_gui.py 28](#_Toc137583855)

[class\_listener.py 33](#_Toc137583856)

[class\_broadcast.py 35](#_Toc137583857)

[utils.py 39](#_Toc137583858)

[Db\_manager.py 45](#_Toc137583859)

[chat\_page.ui 51](#_Toc137583860)

[Create\_new\_user.ui 59](#_Toc137583861)

[Create\_room\_page.ui 63](#_Toc137583862)

[Login\_page.ui 66](#_Toc137583863)

[Main\_page.ui 68](#_Toc137583864)

[not\_a\_guide\_buy\_license.ui 72](#_Toc137583865)

[Style.css 74](#_Toc137583866)

[Chat\_menu.py 75](#_Toc137583867)

[user\_menu.py 78](#_Toc137583868)

## מבוא

### ייזום

הפרוייקט שלי הוא תוכנה בשביל מדריכי טיולים, היא יכולה לתת למדריך ליצור חדר ואז כל מי שנמצע איתו בטיול יוכל להקשיב לו דרך התוכנה כשהוא נכנס לחדר, זה שימושי במקום להשתמש במקרופון או בווקי טוקי לכל אחד שעלול להישבר או ללכת לאיבוד. בחרתי בפרוייקט הזה בגלל שכשהייתי בסיור ביד ושם המדריכה נתנה ווקי טוקי לכל אחד עם אוזניות חד פעמיות וזה היה לט נוח לשימוש וכנראה שגם ישן, בנוסף אחד מהווקי טוקי כמעט אבד וזה גם בעיה. האתגרים שיש הם: לעשות שזה יעבוד בכל הפלטפורמות והטלפונים וגם בווב, לעשות שזה יהיה נגיש לכולם

האפליקציה תהיה מיועדת בעיקר למדריכי טיולים אך יהיה אפשר להשתמש בה גם בכיתות או במקומות עם הרבה אנשים שצריך להקשיב למישהו אחד

המטרה שלי זה שזה יהיה מספיק נוח כדי שכל אחד יוכל להשתמש בזה בלי בעיה ושזה יהיה בשימוש על ידי קשת רחבה של משתמשים

התועלת של זה שזה חסכוני בכסף ושלא צריך לקנות מכשיר בשביל שכולם יוכלו לשמוע

הטכנולוגיה קיימת אך לא לשימוש כזה, יש שיחות טלפון רגילות אך זה לא נוח לעשות שיחה משותפת וזה לא נוח לעשות שמישהו אחד מדבר, יש זום אבל זה כבד מדי לשימוש פשוט וזולל הרבה בטרייה והוא גם לא מספיק קל לשימוש וצריך לשלם בשביל חדר שלא נסגר, יש גם תחליפים לזום אבל יש להם את אותם בעיות של זום

הפרויקט קשור בעיקר לרשתות ותקשורת בין משתמשים וגם בין פלטפורמות ומערכות הפעלה שונות

#### הגדרת לקוח:

יש שתי סוגים של לקוחות: מדריכים (או מורים) שפותחים את החדר ומדברים, ומשתמשים רגילים שהולכים עם המדריכים (או מורים) ונכנסים לחדר שלהם ומקשיבים למה שהם אומרים

#### הגדרת יעדים/מטרות:

לעשות שזה יהיה כמה שיותר נוח לשימוש ונגיש לכמה שיותר אנשים אבל להשאיר שזה יהיה תוכנה קלה

#### בעיות תועלות וחסכונות:

התועלת של התוכנה שלי זה שהיא נוחה למדריכים (או מורים) שרוצים להדריך לקבוצה של אנשים שצריכים להקשיב להם והם נמצאים המרחק מה מהם וקשה להקשיב, עוד תועלת זה שזה יהיה מוכן למדריכים בעיקר עם פונקציות שהם צריכים כמו למשל שיתוף מיקום למקרה שמישהו הלך לאיבוד

#### סקירת פתרונות קיימים:

Zoom – זה טוב לשיחות אבל יש זמן מוגבל לחדר אם לא משלמים, האפליקציה לא כל כך אינטואיטיבית והיא זוללת סוללה ומאוד כבדה, בנוסף היא מיועדת בעיקר ללמידה ולא לטיולים

יש עוד תוכנות טיולים אבל זה בעיקר לתכנון מסלולים ולא למדריכים

#### סקירת טכנולוגית הפרויקט:

צריך מקרופון, רמקול וgps למשתמשים, אין טכנולוגיה מיוחדת

#### תיחום הפרויקט:

יש שירות צד לקוח בשביל להתחבר לחדרים וליצור משתמשים

יש שירות peer to peer בשביל השיחות

יש שירות במערכות הפעלה בשביל להשמיע קול, להשתמש בהקלטה וgps

### פירוט תיאור המערכת (אפיון)

במערכת יהיה אפשר ליצור חדר, זה יצור קוד וכל משתמש שיכניס את הקוד יכנס לחדר ורק מי שיצר את החדר יוכל לדבר ויהיה לו גישה של admin, יהיה אפשר לשוחח בצאט והadmin יוכל להביא לאנשים לדבר או להעיף משתמשים ולנעול את החדר, יהיה גם text to speech שיכתוב את מה שנאמר (טוב לכבדי שמיעה) וזה יתרגם גם לשפות אחרות בגוגל תרגום, יהיה אפשר גם לשתף מקום ולסמן מקומות למקרה שמישהו הולך לאיבוד או שעושים סיבוב בשוק ואז רוצים להיפגש שוב בנקודה מסויימת

יהיה משתמש admin, ומשתמשים רגילים שלפעמים יהיה להם פריווילגיות מסוימות

לסיים את הgui, לסיים את האפשרות לייצור שיחה, לעשות שרת שאחראי על שליחת הקודים וחיבור לחדרים ויצירת משתמשים, לחבר הכל ביחד, להוסיף את שאר הפיצרים

יש סיכונים שיהיה אפשר להתחבר לחדרים בלי שיודעים אז צריך לעשות שלא יהיה קל לפצח את הקוד, יש סיכון שיצותתו גם אז להצפין את השיחה, יש אפשרות שיגנבו מידע על משתמשים אז להצפין את הdb

## תיאור תחום הידע

### פירוט מעמיק של היכולות שהוצגו בשלב הקודם

צד שרת צריך ליצור חדרים וליצור משתמשים מסוג רגיל ומסוג guides

שם היכולת: ליצור חדרים

מהות: ליצור חדר שבו יהיה אפשר לדבר

אוסף יכולות: לקבל את המשתמש שפותח את החדר ולבדוק אם הוא מסוג guide, אם כן אז ליצור קוד שלא היה קיים לפני זה ולשמור בdatabase עם הip שלו כדי שיהיה אפשר להתחבר

שם היכולת: ליצור משתמשים

מהות: ליצור משתמשים משתי סוגים: רגיל ומדריך (לא חובה להתחבר בתור משתמש רגיל, יהיה אפשר גם להתחבר אנונימית)

אוסף יכולות: לקבל שם משתמש, סיסמה ומייל, לבדוק שלא השתמשו במייל הזה לפני, צריך גם לשמור בdatabase ולתת את הסוג המתאים

שם היכולת: לקבל ולשלוח ip וport של מדריך על פי הקוד

מהות: לקבל ולשלוח ip וport בשביל שיהיה אפשר להתחבר לשיחה

אוסף יכולות: לקבל קוד של חדר כלשהו, לעשות שאילתה לדאטה בייס ולשלוח ip וport של המדריך

שם היכולת: לפתוח חדר חדש

מהות: לקבל בקשה לפתיחת חדר אם הוא מדריך

אוסף יכולות: לבדוק אם הוא מדריך, להוסיף את הקוד לטבלה של החדרים הפועלים, לקבל את הip והport של המדריך ולהכניס גם אותם לטבלה

צד לקוח צריך לדעת לקבל את השיחה ולהשמיע, לשלוח את הקול לכולם אם הוא מדבר, לקבל ולשלוח הודעות ומיקום

שם היכולת: לקבל את השיחה ולהשמיע

מהות: לקבל את השיחה ולהשמיע

אוסף יכולות: לקבל את השיחה ולהשמיע

שם היכולת: לשלוח את הקול לכולם אם הוא מדבר

מהות: לשלוח את הקול לכולם אם הוא מדבר

אוסף יכולות: להקליט ולשלוח

שם היכולת: לקבל ולשלוח הודעות

מהות: לקבל ולשלוח הודעות

אוסף יכולות: לקבל הודעות ולשלוח הודעות ולהראות מי שלח מה ומתי

שם היכולת: לקבל ולשלוח מיקום

מהות: לקבל ולשלוח מיקום

אוסף יכולות: לקבל את המיקום של מי שמשתף את המיקום שלו ולהראות איפה הוא על המפה, לשלוח לכולם את המיקום כשהוא רוצה

אובייקטים נחוצים: ממשק משתמש, הצפנה, תקשורת, שרת, בסיס נתונים

## מבנה / ארכיטקטורה של הפרויקט

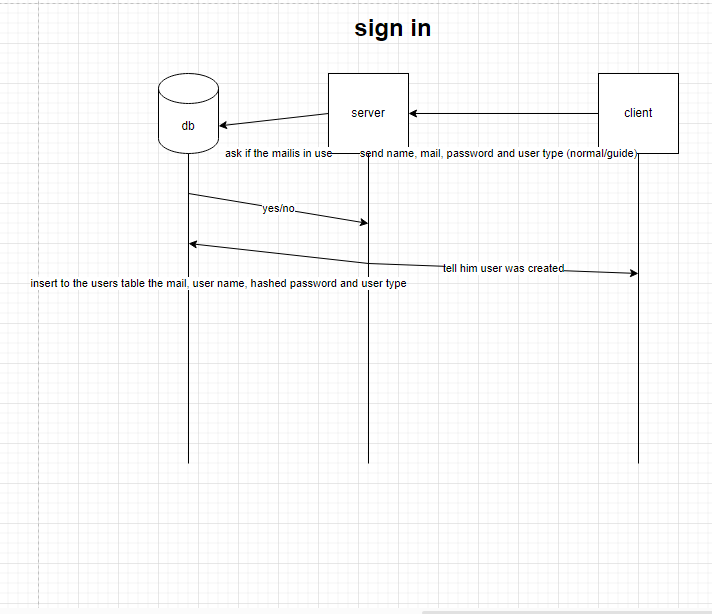
1. תיאור הארכיטקטורה של המערכת המוצעת:

למערכת המוצעת יש ארכיטקטורת שרת-לקוח וארכיטקטורתpeer to peer , יצירת המשתמשים נעשית בין המשתמש לשרת, החדרים גם נעשים בין המשתמש לשרת, השרת שומר את כל החדרים הפתוחים ומתאים בין הקוד שהמשתמש מכניס לבין המדריך שפתח את החדר ונותן לו את הip והport הפתוחים כדי שיהיה אפשר לתקשר ביניהם. אפליקציית המדריך פועלת כשרת ואפליקציות המבקרים פועלות כלקוחות בשיטת peer to peer. המדריך יוצר חדר ומקבל קוד ייחודי בו יוכלו המבקרים להשתמש כדי להצטרף לחדר. המבקרים מתחברים לחדר באמצעות הקוד שנשלח לשרת והם מקבלים את הip והport. המדריך יכול לדבר דרך התוכנה. התוכנה משתמשת בפרוטוקול UDP לתקשורת בין המדריך למבקרים כדי שיהיה תקשורת מהירה גם אם חלק מהמידע לא עבר טוב. המערכת משתמשת גם במסד נתונים מקומי של SQLite לאחסון המשתמשים והחדרים הפתוחים.

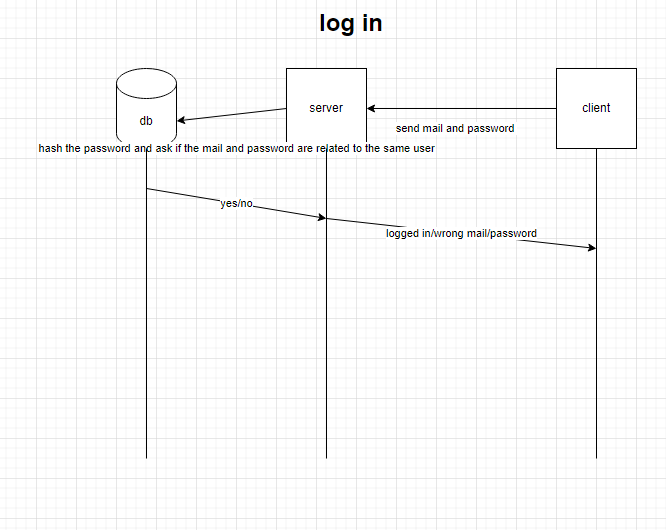
תיאור החומרה:

המערכת מיושמת באמצעות Python כשפת התכנות העיקרית. PySide6 משמש לבניית ממשק המשתמש. Sockets ופרוטוקול UDP משמשים לתקשורת בין המדריך למשתמש וללקוח. PyAudio משמש להקלטה והשמעת השיחה. SQLite משמש לאחסון משתמשים וחדרים.

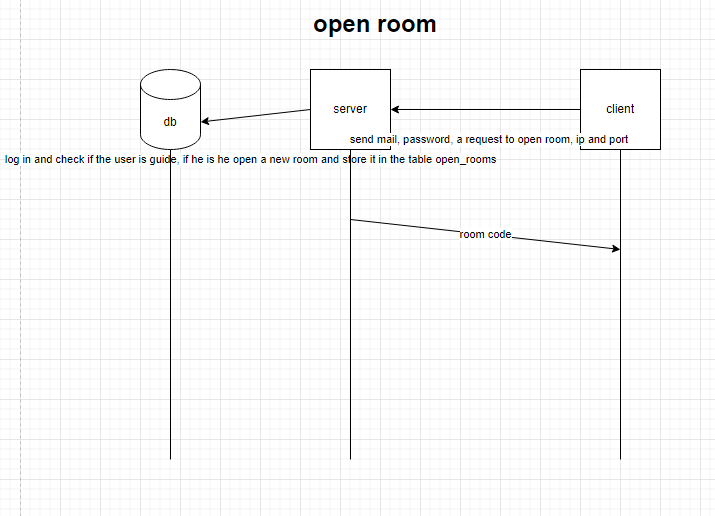
## Sign in



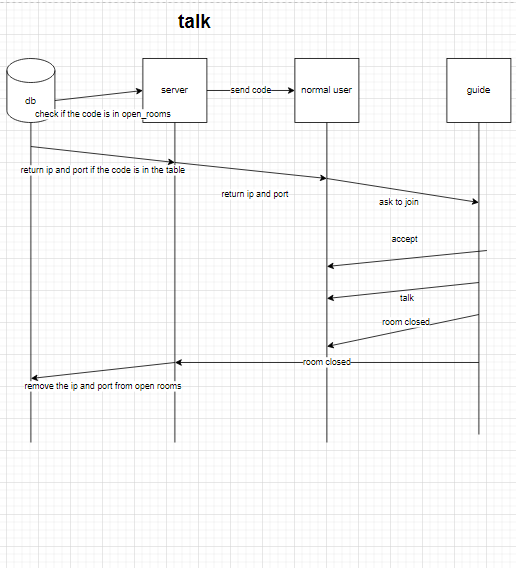
Log in



Open room



Talk



2. תיאור הטכנולוגיה הרלוונטית:

שפת תכנות: פייתון

פריימוורק לגרפיקה: pyside שמשתמש בqt לייצירת גרפיקה קרוס פלטפורם וcss לעיצוב העמודים

תקשורת: socket בשימוש פרוטוקול udp כדי שיהיה סטרים מהיר של מידע בזמן אמת, השתמשתי בjson כדי להעביר את המידע

התוכנה מיועדת לשימוש במוזיאונים כי שם יש רשת סגורה עם קליטה טובה ואנשים בדרך כלל רוצים להסתובב קצת מחוץ לטווח השמיעה של המדריך אבל אפשר להשתמש גם בטיולים בחוץ ובכיתות למקרה שהתלמידים בעבודת כיתה רועשת

3. תיאור זרימת המידע במערכת:

ניתן לחלק את זרימת המידע במערכת לשלוש יכולות עיקריות:

- יצירת חדר: המדריך יוצר חדר ומקבל קוד ייחודי. הקוד נשלח למבקרים.

- הצטרפות מבקרים: המבקרים מתחברים לחדר באמצעות הקוד שמספק המדריך.

- צ'אט: המדריך יכול לדבר למשתמשים, והמשתמשים יכולים לשלוח הודעות טקסט למדריך.

תמונה שמכילה טקסט, תרשים, קו, מקביל

התיאור נוצר באופן אוטומטי

4. תיאור האלגוריתמים המרכזיים בפרויקט:

שרת:

תהליך הכניסה דורש מהמשתמש לספק את המייל והסיסמה שלו, אשר נשלחים לאחר מכן לשרת לאימות. השרת בודק במסד נתונים את המשתמשים, ואם המשתמש בטבלה השרת מעניק למשתמש גישה למערכת. אם המייל והסיסמה לא תואמים או לא במסד נתונים, השרת ימנע גישה ויבקש מהמשתמש לנסות שוב.

תהליך ההרשמה מקבל את השם וכתובת האימייל של המשתמש וסיסמה ואת סוג המשתמש (מדריך או משתמש). לאחר מכן השרת בודק אם המייל בשימוש ואם לא אז הוא מאחסן את המידע במסד הנתונים ומעניק למשתמש גישה למערכת.

השרת מקשיב לפקודות, אם הפקודה זה "פתח חדר", השרת בודק תחילה אם המשתמש מורשה לפתוח חדר על ידי בדיקה אם המשתמש הוא מדריך, הקוד מקבל את ה-IP וה-port של המדריך, ואז מכניס לטבלה את החדר הפתוח עם ה-IP וה-port ומקשר למדריך ומחזיר את הקוד של החדר.

אם הפקודה זה "סגור חדר", השרת בודק תחילה אם המשתמש מורשה לפתוח חדר על ידי בדיקה אם המשתמש הוא מדריך ואם המשתמש מדריך, הקוד מוחק את נתוני החדר המתאימים ממסד הנתונים.

אם הפקודה היא "קבל מארח לפי קוד", הקוד מחלץ את קוד החדר מההודעה, בודק את נתוני ה-IP וה-port המתאימים של המדריך ממסד הנתונים, ושולח את הנתונים ללקוח.

מדריך:

המדריך מקשיב לכל הודעה ותמיד שולח לכל מי שמחובר את ההקלטה בזמן אמת, אם הוא מזהה שמישהו נכנס הוא שומר את הip והport והשם שלו ברשימה של האנשים שיש בחדר.

אם הוא מקבל שמישהו שולח הודעה הוא מקבל את תוכן ההודעה ומי ששלח ושולח לשאר המאזינים מי שלח את ההודעה ומה תוכן ההודעה.

לקוח:

הלקוח שולח בקשה לip וport שהוא קיבל מהשרת כששלח את הקוד.

כשהמדריך מאשר לו להיכנס הוא תמיד מקשיב למדריך. אם זה שמע הוא משמיע ואם זה json הוא בודק מה התוכן ועושה, בדרך כלל זה הודעה או קריאה לסגירת החדר.

5. תיאור סביבת הפיתוח:

סביבת הפיתוח לפרויקט זה כוללת כלים שונים, כגון Python, PySide6, PyAudio ו-SQLite. הבדיקה נעשית בעיקר בשרת שמשתמש במספר טכניקות אבטחה כדי להבטיח את בטיחות נתוני המשתמש.

ראשית, הקוד משתמש ב-Salted Password hashing כדי לאחסן את הסיסמה של המשתמש בצורה מאובטחת. כאשר משתמש יוצר חשבון, הסיסמה שלו עוברת גיבוב עם רעש (רצף אקראי של בייתים) באמצעות האלגוריתם SHA256. הרעש מתווסף לסיסמה כדי להפוך את ה-hash לבטוח יותר, מכיוון שאותה סיסמה תייצר גרסאות שונות עם רעשים שונים. לאחר מכן ה-hash המתקבל מאוחסן במסד הנתונים, במקום בסיסמה הפשוטה. כאשר בודקים את הסיסמה מצפינים מחדש את הסיסמה ומשווים את הhash ככה אם פורצים למסד נתונים אי אפשר להשתמש בסיסמה או לדעת מה היא הייתה.

שנית, הקוד מאמת את כתובת האימייל של המשתמש לפני יצירת חשבון. כתובת הדוא"ל נבדקת מול ביטוי רגולרי כדי לוודא שהיא בפורמט חוקי. זה עוזר למנוע מתוקפים להחדיר קוד או נתונים זדוניים למסד הנתונים.

שלישית, הקוד משתמש בהצהרות מוכנות כדי למנוע התקפות של הזרקת SQL. הצהרות מוכנות משמשות להפרדה בין קוד SQL לבין קלט המשתמש. זה מונע מתוקפים להחדיר קוד SQL משלהם למסד הנתונים על ידי שינוי קלט המשתמש. בנוסף הקוד משתמש בregex בשביל לבדוק אם יש sql injection.

רביעית, השרת מתקשר בjson ולא בpickle שיש לו בעיות אבטחה כאשר משתמשים בו. זה יכול להפעיל קוד במהלך תהליך הדה-סריאליזציה וניתן להשתמש בו עבור התקפות דה-סריאליזציה.

6. תיאור פרוטוקול התקשורת:

פרוטוקול התקשורת כולל שליחת וקבלת הודעות בין המדריך למבקרים באמצעות פרוטוקול UDP. מבנה ההודעות כולל שדה סוג הודעה, שדה שולח, שדה נמען ושדה תוכן הודעה. ההודעות נשלחות בפורמט בינארי שכשהופכים לסטרינג יש הודעת json שאפשר להפוך לdictionary בעזרת הספרייה json של פייתון.

7. תיאור מסכי המערכת:

המערכת כוללת מסכים שונים כמו מסך הראשי שבו אפשר גם לשים קוד ולהתחבר לחדר, מסך ההרשמה, מסך התחברות, מסך יצירת החדר ומסך השיחה.

8. תיאור מבני הנתונים:

Users:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | username | password | email | type | room\_code | host\_ip | host\_port |
| 1 | user1 | gO2v0JW1B | [us1@gmail.com](mailto:user1@example.com) | normal | NULL | NULL | NULL |
| 2 | user2 | hsg9yQVPjW | [us2@gmail.com](mailto:user2@example.com) | guide | A1B2C3 | 192.168.0.1 | 8080 |
| 3 | user3 | P7hEKxrn4J1 | [us3@gmail.com](mailto:user3@example.com) | normal | NULL | NULL | NULL |

Active\_rooms:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| id | room\_code | host\_ip | host\_port | host\_listener\_port |
| 1 | A1B2C3 | 192.168.0.1 | 8080 | NULL |

9. סקירת חולשות והאיומים:

איומים אפשריים למערכת כוללים התקפות Injection SQL, התקפות אתרים, התקפות MITM, והתקפות DDOS.

## מימוש הפרויקט

חלק א:

socket: מודול זה מספק דרך ליצור שקעים ולקיים אינטראקציה עם שקעים.

threading: מודול זה מספק אפשרות להריץ חלקים מהקוד במקביל.

sqlite3: מודול זה מספק דרך ליצור אינטראקציה עם מסד נתונים של SQLite.

hashlib: מודול זה מספק דרך ליצור ולנהל hashs.

hmac: מודול זה מספק דרך ליצור ולנהל HMACs.

json: מודול זה מספק דרך ליצור ולנתח נתוני JSON.

מודולים שפיתחתי:

utils.py

מחלקה: לממש כל מיני פונקציות ומחלקות שצריך במהלך הפרויקט.

תפקיד: מספק פונקציות ומחלקות עבור המערכת.

פעולות במחלקה:

create\_room\_id()

תפקיד: יוצר מזהה ייחודי לחדר.

פרמטרים: אין

הצהרת החזרה: קוד לחדר.

send\_response()

תפקיד: שולח תגובה ללקוח.

פרמטרים: נתוני התגובה וכתובת הלקוח.

הצהרת החזרה: אין.

compare\_passwords()

תפקיד: השוואה בין שתי סיסמאות.

פרמטרים: שתי הסיסמאות.

הצהרת החזרה: נכון אם הסיסמאות תואמות, אחרת לא נכון.

התחברות()

תפקיד: כניסה למשתמש.

פרמטרים: האימייל והסיסמה של המשתמש.

הצהרת החזרה: מזהה המשתמש אם הכניסה מוצלחת, אין אחרת.

get\_room\_code()

תפקיד: מקבל את קוד החדר עבור משתמש.

פרמטרים: מזהה המשתמש.

הצהרת החזרה: קוד החדר למשתמש, אין אם למשתמש אין חדר.

insert\_room()

תפקיד: הוספת חדר למסד הנתונים.

פרמטרים: נתוני החדר.

הצהרת החזרה: אין.

get\_user\_type()

תפקיד: מקבל את סוג המשתמש עבור משתמש.

פרמטרים: מזהה המשתמש.

הצהרת החזרה: סוג המשתמש עבור המשתמש, אין אם המשתמש אינו קיים.

יחסי גומלין:

מודול utils.py משמש את כל שאר התוכנה. מודול server.py הוא המודול הראשי של המערכת והוא מטפל בחיבורים נכנסים מלקוחות. מודול utils.py משמש גם את מודול server.py לשליחת תגובות ללקוחות ולהשוואת סיסמאות. client\_gui.py משתמש ב class\_broadcast.py וב class\_listener.py בשביל להשתמש בתקשורת של החדרים והם גם משתמשים בutils.py והם גם מתקשרים עם הserver.py

חלק ב:

def compare\_passwords(password:bytes, hashed\_password:bytes, salt:bytes):

"""This function compares two passwords.

Args:

password: The first password.

hashed\_password: The hashed version of the second password.

salt: The salt used to hash the second password.

Returns:

True if the passwords match, False otherwise.

"""

# Hash the first password with the same salt.

salted\_password = hashlib.sha256(password + salt).hexdigest().encode()

# Compare the hashed passwords.

return hmac.compare\_digest(salted\_password, hashed\_password)

הפונקציה compare\_passwords() פועלת על ידי הצפנת הסיסמה הראשונה עם אותו רעש ששימש להצפנת הסיסמה השנייה. לאחר מכן הפונקציה משווה את הסיסמאות המוצפנות. אם הסיסמאות תואמות, הפונקציה מחזירה True. אחרת, הפונקציה מחזירה False.

הפונקציה compare\_passwords() מסובכת מכיוון שהיא משתמשת בשני אלגוריתמי הצפנה שונים: hashlib.sha256() ו-hmac.compare\_digest(). האלגוריתם hashlib.sha256() משמש לגיבוב הסיסמאות, בעוד שהאלגוריתם hmac.compare\_digest() משווה את הסיסמאות המוצפנות באותו כמות זמן אם הסיסמה נכונה או לא נכונה.

def receive\_data():

"""This function receives data from the socket and starts a new thread to handle the connection.

Args:

sock: The socket.

Returns:

None.

"""

# Receive data from the socket.

data = b""

while True:

chunk, addr = sock.recvfrom(1024 \* 2)

data += chunk

if len(chunk) < 1024:

# End of message.

break

# Print the data.

print(data, "hi")

# Start a new thread to handle the connection.

threading.Thread(target=handle\_connection, args=(data, addr)).start()

הפונקציה receive\_data() על הזמן מקשיבה לחיבורים ולאחר מכן משרשרת להודעה ובודקת אם זה כל ההודעה שנשלחה, אם לא היא ממשיכה להקשיב עד לקבלת ההודעה המלאה. הפונקציה קוראת לhandle\_connection() שתעבד את הנתונים על thread אחר.

הפונקציה receive\_data() מסובכת מכיוון שהיא משתמשת בשתי טכניקות שונות לטיפול בנתונים: היא משתמשת בלולאה כדי לקבל את הנתונים והיא משתמשת בשרשור כדי לעבד את הנתונים. הלולאה משמשת לקבלת הנתונים מכיוון שהיא לפעמים קיבלה רק חלק מהנתונים ואז השרת לא ידע לעבוד איתם.

sql\_injection\_pattern = re.compile(r"(?:')|(?:--)|(\b(select|update|delete|insert|drop|alter)\b)")

הקוד בודק אם יש sql injection

def send\_udp\_large\_data(sock, data, buffer\_size=1024):

"""This function sends a large amount of data over UDP.

Args:

sock: The socket to send the data over.

data: The data to send.

buffer\_size: The size of each packet to send.

Returns:

None.

"""

# Calculate the number of packets required to send the entire data.

packet\_count = len(data) // buffer\_size

if len(data) % buffer\_size != 0:

packet\_count += 1

# Send the data in pieces.

for i in range(packet\_count):

# Get the current piece of data to send.

start = i \* buffer\_size

end = min(start + buffer\_size, len(data))

data\_chunk = data[start:end]

# Send the current piece of data to the server.

sock.sendto(data\_chunk, (SERVER\_IP, SERVER\_PORT))

הקוד פועל על ידי חישוב תחילה של מספר החבילות שידרשו לשליחת כל הנתונים. זה נעשה על ידי חלוקת אורך הנתונים בגודל המאגר. אם אורך הנתונים אינו כפולה של גודל המאגר, אזי תידרש מנה נוספת.

לאחר שחושב מספר החבילות, הקוד שולח את הנתונים בחתיכות. פיסת הנתונים הראשונה נשלחת על ידי קבלת פיסת הנתונים הנוכחית לשליחה, מה שנעשה על ידי קבלת האינדקס של הביט הראשון בנתונים והאינדקס של הבית האחרון בנתונים. פיסת הנתונים הנוכחית נשלחת לאחר מכן לשרת באמצעות שיטת sock.sendto() .

לאחר מכן הקוד ממשיך לשלוח את הנתונים בחתיכות עד שכל הנתונים נשלחים.

חלק ג:

sql\_injection\_pattern

מטרה: לבדוק התקפות הזרקת SQL.

מה הוא עושה: הביטוי הרגולרי מחפש התאמה לדפוס ספציפי של תווים שיכול לשמש להחדרת קוד זדוני לשאילתת SQL.

תוצאות: אם נמצא התאמה, הפונקציה תעלה חריגה.

email\_regex

מטרה: לבדוק אם כתובת דוא"ל תקפה.

מה הוא עושה: הביטוי הרגולרי מחפש התאמה לתבנית ספציפית של תווים שתקפים בכתובת דוא"ל.

תוצאות: אם נמצאה התאמה, הפונקציה מחזירה True. אם לא נמצא התאמה, הפונקציה מחזירה False.

login()

מטרה: לבדוק אם סיסמת המשתמש נכונה.

מה היא עושה: הפונקציה לוקחת שני ארגומנטים: סיסמת המשתמש והסיסמה הגובבת המאוחסנת במסד הנתונים. הפונקציה משווה את הסיסמה של המשתמש לסיסמת הגיבוב.

תוצאות: אם הסיסמאות תואמות, הפונקציה מחזירה את מזהה המשתמש. אם הסיסמאות אינן תואמות, הפונקציה מחזירה ללא.

compare\_passwords()

מטרה: להשוות סיסמה של משתמש לסיסמה הגובבת המאוחסנת במסד הנתונים.

מה היא עושה: הפונקציה לוקחת שני ארגומנטים: סיסמת המשתמש והסיסמה הגובבת המאוחסנת במסד הנתונים. הפונקציה משתמשת בפונקציית hash קריפטוגרפית כדי להמיר את סיסמת המשתמש לסיסמה hashed. לאחר מכן, הפונקציה משווה את הסיסמה הגובבת לסיסמת הגיבוב המאוחסנת במסד הנתונים.

תוצאות: אם הסיסמאות הגובבות תואמות, הפונקציה מחזירה True. אם הסיסמאות הגובבות אינן תואמות, הפונקציה מחזירה False.

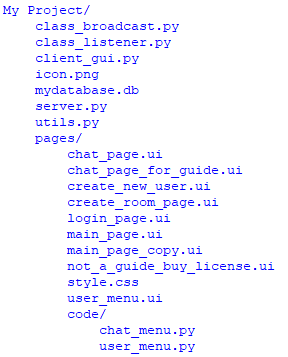
get\_room\_code()

מטרה: לבדוק אם למשתמש יש קוד חדר.

מה היא עושה: הפונקציה לוקחת ארגומנט אחד: מזהה המשתמש. הפונקציה מבצעת שאילתות במסד הנתונים כדי לראות אם למשתמש יש קוד חדר.

תוצאות: אם למשתמש יש קוד חדר, הפונקציה מחזירה את קוד החדר. אם למשתמש אין קוד חדר, הפונקציה מחזירה ללא.

## מדריך למשתמש

עץ קבצים:  


התקנת מערכת:

צריך את התוכנה, חיבור לאוזניות ומיקרופון וחיבור לאינטרנט

בשביל להשתמש צריך לפחות משתמש אחד מסוג מדריך שיפתח את החדר והוא חייב מייל בשביל זה, כל השאר יכולים להיות אנונימיים

משתמשי המערכת:

יש שתי סוכי משתמשים: מדריך ונורמאלי

מדריך:

הוא יכול לפתוח ולנהל חדרים

הוא צריך לפתוח חדר ואז להראות למשתמשים את הקוד של החדר כדי שיכנסו, שם הוא יכול לדבר

תמונה שמכילה טקסט, צילום מסך, קו, גופן

התיאור נוצר באופן אוטומטי

בעמוד הזה הוא יכול לפתוח את החדר או לבקש לשנות את הקוד

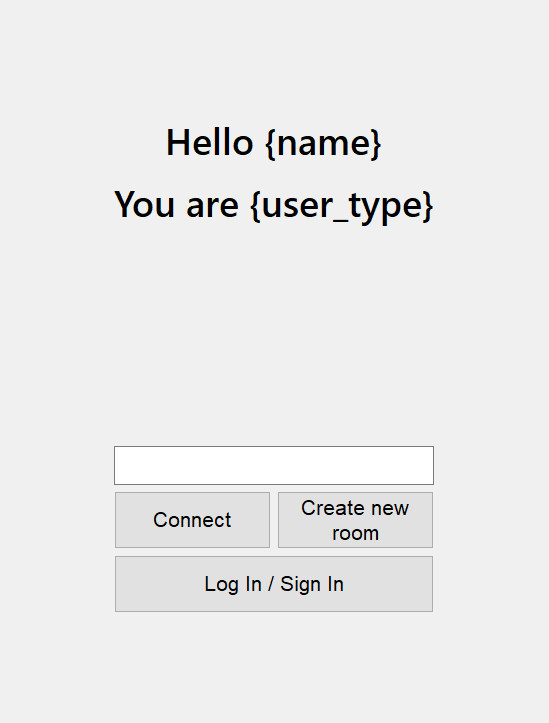
תמונה שמכילה טקסט, צילום מסך, גופן, עיצוב

התיאור נוצר באופן אוטומטי

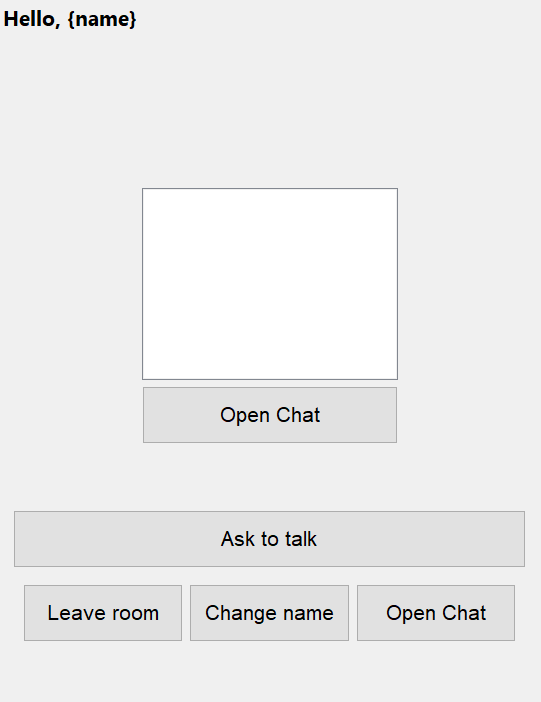
כאן המדריך יכול לשלוט בחדר ולדבר

משתמש רגיל:

הוא יכול להתחבר לחדרים בעזרת הקוד ולהתכתב



כאן המשתמש יכול להתחבר, ליצור משתמש, לפתוח חדר (אם הוא מדריך) ולהתחבר לחדרים



כאן המשתמש יכול לשמוע את המדריך ולשלוח הודעות

## סיכום אישי / רפלקציה

בהתחלה למדתי איך להעביר את ההקלטה בזמן אמת ולמדתי על הספרייה הגרפית שאני משתמש בה, באותו זמן התחלתי לעבוד על המסד נתונים ועל השרת בגלל שזה דברים שאני יודע כבר איך לעשות. אחר כך התמקדתי בלהעביר את ההקלטה בזמן אמת והיו לי הרבה מאוד בעיות עם זה, בחלק מהמקרים שמעו לא טוב, בחלק מהמקרים שמעו בהילוך מהיר ואיטי ובחלק היו שומעים בצורה מוזרה, בדקתי את זה הרבה מאוד פעמים ולמדתי הרבה על איך שמע והקלטות עובדות במחשב עד שהבנתי אחרי הרבה מאוד זמן שהבעיה הייתה באוזניות שלי שהיו מאוד ישנות, כשבדקתי עם אוזניות חדשות הכל עבד נהדר אז אזהרה למשתמשים: **אל תשתמשו באוזניות ישנות או ברמקולים של המחשב, זה עובד נוראי וזה עלול לעשות רעשים חזקים ולא צפויים!**

אחרי שהיה לי תקשורת עובדת עבדתי על המסכים והגרפיקה, שזה החלק הפחות אהוב עלי, כל פעם שחשבתי שסיימתי הבנתי שלא תכננתי עד הסוף ושחסרים לי כפתורים כל פעם, הפיתרון שלי היה להשאיר את זה לסוף, אחר כך הייתי צריך לעצב שזה יראה איכשהו נורמאלי ויהיה נחמד להסתכל על זה ושזה לא יהיה סתם הסגנון הדיפולתי של המערכת אז עבדתי כמה שעות על העיצוב והצבעים.

בסוף חיברתי הכל אחד לשני בצורה שיעבוד והפכתי חלק מהקוד למחלקות כדי שיהיה קל יותר "להלביש" את זה על הגרפיקה.

למדתי הרבה על איך תקשורת udp עובדת, איך שמע עובד במחשב ועל דרכים שונות ליצור גרפיקה בפייתון.

הכלים שנשארים איתי להמשך הם איך ללמוד ולנהל זמן, טריקים לעיצוב, ידע כללי על שמע והקלטה במחשבים וגם ידע מעמיק יותר על תקשורת udp

אם הייתי צריך לעשות הכל מחדש הייתי מנסה לבחור דרך אחרת להעביר את השמע, בחרתי בספרייה שהייתה low level כי חשבתי שזה יהיה נוח בשבילי ללמוד, וזה היה טעות כי יש ספריות שעוזרות עם תקשורת והן עושות הרבה דברים שאני הייתי צריך ליצור מ-0 וזה לקח הרבה מאוד זמן

אם היו לי משאבים נוספים הייתי משפר את הגרפיקה כי היא לא משהו ולא הכי נוח להשתמש וה-ux לא אופטימלי, הייתי מוסיף עוד אופציות שליטה למדריך (כמו אפשרות לאשר לחדר)

תודות: אני הכי רוצה להודות לאבא שלי שעזר לי עם רעיונות לפרויקט ועזר לי אם איך הפרויקט אמור להיראות ועל מה לחשוב, אני רוצה להודות גם לירין והראל זאבי שעזרו לי בגרפיקה והתייעצתי איתם, אני רוצה להודות גם לאיתי שרון שהתייעצתי איתו על כל מיני דברים שממש עזרו לי ואני רוצה להודות גם למורה שרית שהכווינה אותי בפרוייקט ועזרה לי לראות איך הפרוייקט אמור להיראות בתור מוצר מוכן ולא כמו שחשבתי מקודם שאני פשוט אצור משהו ואוסיף עוד פונקציות תוך כדי

## ביבליוגרפיה

גרפיקה:

<https://www.pythonguis.com/tutorials/pyside-creating-your-first-window/>

<https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QStackedWidget.html>

תקשורת:

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

<https://www.youtube.com/watch?v=3qlhbez-RPI>

הקלטה:

<https://people.csail.mit.edu/hubert/pyaudio/docs/>

# קוד

## Server.py

import socket

import threading

import sqlite3

import hashlib

import hmac

import json

import json

import re

import utils

"""

todo list:

finishe the check for sql injections

"""

salt = b'your\_salt'

# Function to handle incoming connections

def handle\_connection(data, addr):

# Unjson the data

data = json.loads(data.decode(), strict=False)

print(data, addr)

sql\_injection\_pattern = re.compile(r"(?:')|(?:--)|(\b(select|update|delete|insert|drop|alter)\b)")

"""

Doesn't work, check later

# check for SQL injection

if sql\_injection\_pattern.search(str(data)):

send\_response(addr, {"message": "Invalid command.", "code": 400})

print("Possible SQL injection detected!")

"""

# example {"command": "create user", "data": {"username": "first2", "password": "123", "email": "check6@gmail.com", "type": "guide"}}

# Check the command

if data["command"] == "create user":

# Get the user data

user\_data = data["data"]

# Encrypt the password

password = user\_data["password"].encode()

salted\_password = hashlib.sha256(password + salt).hexdigest()

# Check if email is valid

email\_regex = '^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

if re.search(email\_regex, user\_data["email"]):

# Connect to the database

connection = sqlite3.connect("mydatabase.db")

cursor = connection.cursor()

# Check if the email is already in use

cursor.execute("SELECT \* FROM Users WHERE email=?", (user\_data["email"],))

result = cursor.fetchone()

if result:

send\_response(addr, {"message": "Email already in use.", "code": 406})

print("Email already in use.")

else:

# Insert the data into the users table

if user\_data["type"] == 'normal':

# Insert the data into the users table

cursor.execute("INSERT INTO users (username, password, email, type) VALUES (?, ?, ?, ?)", (user\_data["username"], salted\_password, user\_data["email"], user\_data["type"]))

send\_response(addr, {"message": "User created successfully.", "code": 201})

print("User created successfully.")

elif user\_data["type"] == 'guide':

# Insert the data into the users table

room\_code = utils.create\_room\_id()

cursor.execute("INSERT INTO users (username, password, email, type, room\_code) VALUES (?, ?, ?, ?, ?)", (user\_data["username"], salted\_password, user\_data["email"], user\_data["type"], room\_code))

send\_response(addr, {"message": "User created successfully.", "code": 201, "room code": room\_code})

print("User created successfully.")

print(f"{room\_code = }")

else:

send\_response(addr, {"message": "Incorrect user type.", "code": 406})

print("Incorrect user type.")

connection.commit()

connection.close()

else:

send\_response(addr, {"message": "Invalid command.", "code": 400})

print("Invalid email address.")

elif data["command"] == "login":

# this if is just for check, need to be checked every time

user\_id = login(data["data"]["password"], data["data"]["email"])

if user\_id:

send\_response(addr, {"message": "You logged in.", "code": 201})

print("You logged in.", user\_id)

else:

send\_response(addr, {"message": "Incorrect email or password", "code": 400})

elif data["command"] == "open room":

user\_info = data["user\_info"]

if user\_info == None:

send\_response(addr, {"message": "You are not authorized to open a room.", "code": 403})

return

user\_id = login(user\_info["password"], user\_info["email"])

if user\_id is None:

# Return error message to client

send\_response(addr, {"message": "Invalid password or mail.", "code": 401})

return

user\_type = get\_user\_type(user\_id)

if user\_type != "guide":

# Return error message to client

send\_response(addr, {"message": "You are not authorized to open a room.", "code": 403})

return

room\_code = get\_room\_code(user\_id)

host\_ip = data["data"]["host\_ip"]

host\_port = data["data"]["host\_port"]

host\_listener\_port = data["data"]["host\_listener\_port"]

insert\_room(room\_code, user\_id, host\_ip, host\_port, host\_listener\_port)

# Return success message to client

send\_response(addr, {"message": "Room successfully opened.", "code": 200, "room code": room\_code})

elif data["command"] == "close room":

user\_info = data["user\_info"]

if user\_info == None:

send\_response(addr, {"message": "You are not authorized to open a room.", "code": 403})

return

user\_id = login(user\_info["password"], user\_info["email"])

if user\_id is None:

# Return error message to client

send\_response(addr, {"message": "Invalid password or mail.", "code": 401})

return

user\_type = get\_user\_type(user\_id)

if user\_type != "guide":

# Return error message to client

send\_response(addr, {"message": "You are not authorized to open a room.", "code": 403})

return

with sqlite3.connect("mydatabase.db") as connection:

cursor = connection.cursor()

cursor.execute("DELETE FROM active\_rooms WHERE id=?;", (user\_id, ))

connection.commit()

send\_response(addr, {"message": "Room was closed successfully", "code": 204})

elif data["command"] == "get host by code":

room\_code = data["room\_code"]

with sqlite3.connect("mydatabase.db") as conn:

c = conn.cursor()

c.execute("SELECT host\_ip, host\_port, host\_listener\_port FROM active\_rooms WHERE room\_code = ?", (room\_code, ))

host\_ip, host\_port, host\_listener\_port = c.fetchone()

send\_response(addr, {"message": "Here are the host ip and port", "code": 200, "host ip": host\_ip, "host port": host\_port, "host\_listener\_port": host\_listener\_port})

elif data["command"] == "get type by email":

email = data["email"]

with sqlite3.connect("mydatabase.db") as conn:

c = conn.cursor()

c.execute("SELECT type FROM users WHERE email = ?", (email, ))

user\_type = c.fetchone()

if not(user\_type is None):

user\_type = user\_type[0]

send\_response(addr, {"message": f"The user is {user\_type}", "code": 200, "user type": user\_type})

else:

send\_response(addr, {"message": "Invalid mail.", "code": 401, "user type": None})

elif data["command"] == "get type by name":

email = data["email"]

with sqlite3.connect("mydatabase.db") as conn:

c = conn.cursor()

c.execute("SELECT username FROM users WHERE email = ?", (email, ))

user\_name = c.fetchone()

if not(user\_name is None):

user\_name = user\_name[0]

send\_response(addr, {"message": f"The username is {user\_name}", "code": 200, "user name": user\_name})

else:

send\_response(addr, {"message": "Invalid mail.", "code": 401, "user name": None})

else:

print(data["command"], data["command"] == "open room")

send\_response(addr, {"message": "Invalid command.", "code": 400})

print("Invalid command.")

def send\_response(client, data):

data = json.dumps(data)

sock.sendto(bytes(data, encoding="utf-8"), client)

def compare\_passwords(password:bytes, hashed\_password:bytes, salt:bytes):

print("hi")

salted\_password = hashlib.sha256(password + salt).hexdigest().encode()

print(salted\_password)

print(hashed\_password)

return hmac.compare\_digest(salted\_password, hashed\_password)

def login(password, email):

with sqlite3.connect("mydatabase.db") as conn:

c = conn.cursor()

c.execute("SELECT id, password FROM users WHERE email = ?", (email, ))

user = c.fetchone()

print(user)

if user is not None:

user\_id, hashed\_password = user

print(user\_id, hashed\_password)

if compare\_passwords(str.encode(password), str.encode(hashed\_password), salt):

return user\_id

return None

def get\_room\_code(user\_id):

with sqlite3.connect("mydatabase.db") as connection:

cursor = connection.cursor()

cursor.execute("SELECT room\_code FROM users WHERE id=?", (user\_id,))

room\_code = cursor.fetchone()

if room\_code:

room\_code = room\_code[0]

else:

room\_code = None

return room\_code

def insert\_room(room\_code, user\_id, host\_ip, host\_port, host\_listener\_port):

connection = sqlite3.connect("mydatabase.db")

cursor = connection.cursor()

cursor.execute("INSERT INTO active\_rooms (room\_code, id, host\_ip, host\_port, host\_listener\_port) VALUES (?, ?, ?, ?, ?)", (room\_code, user\_id, host\_ip, host\_port, host\_listener\_port))

connection.commit()

def get\_user\_type(user\_id):

connection = sqlite3.connect("mydatabase.db")

cursor = connection.cursor()

cursor.execute("SELECT type FROM users WHERE id=?", (user\_id,))

user\_type = cursor.fetchone()[0]

connection.close()

return user\_type

# Function to receive data

def receive\_data():

while True:

# Receive data from the socket

# data, addr = sock.recvfrom(1024)

data = b""

while True:

chunk, addr = sock.recvfrom(1024\*2)

data += chunk

if len(chunk) < 1024:

# end of message

break

print(data, "hi")

# Start a new thread to handle the connection

threading.Thread(target=handle\_connection, args=(data, addr)).start()

# Create a socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# set the buffer size to 4096 bytes

sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_RCVBUF, 1048576)

# Bind the socket to the address and port

sock.bind(("localhost", 5005))

# Start the receive data function in a new thread

threading.Thread(target=receive\_data).start()

# Keep the program running

while True:

pass

## client\_gui.py

import sys

import utils

import subprocess

from PySide6.QtGui import QIcon

from PySide6.QtWidgets import QApplication, QMainWindow, QStackedWidget, QLineEdit, QMenu

from PySide6.QtUiTools import QUiLoader

from PySide6.QtCore import QFile, Qt

from pages.code import user\_menu

from pages.code import chat\_menu

import class\_broadcast

import class\_listener

from PySide6 import QtCore

print(QtCore.QCoreApplication.libraryPaths())

# C:\Users\Ariel\AppData\Roaming\Python\Python311\Scripts\pyside6-uic.exe

"""

tasks:

"""

client = utils.Client(None, None, None)

print(client.username)

voice\_client = None

global\_room\_code = None

app = QApplication(sys.argv)

with open("pages/style.css", "r") as f:

app.setStyleSheet(f.read())

app.setWindowIcon(QIcon('icon.png'))

loader = QUiLoader()

main\_page = QFile("pages/main\_page.ui")

main\_page.open(QFile.ReadOnly)

my\_form = loader.load(main\_page)

main\_page.close()

main\_page\_heading\_text = "Hello {name}\nYou are {user\_type}"

create\_room\_page = QFile("pages/create\_room\_page.ui")

create\_room\_page.open(QFile.ReadOnly)

my\_form2 = loader.load(create\_room\_page)

create\_room\_page.close()

create\_room\_page\_heading\_text = "Hello {name}"

chat\_page = QFile("pages/chat\_page.ui")

chat\_page.open(QFile.ReadOnly)

my\_form3 = loader.load(chat\_page)

chat\_page.close()

chat\_page\_heading\_text = "Hello, {name}"

login\_page = QFile("pages/login\_page.ui")

login\_page.open(QFile.ReadOnly)

my\_form4 = loader.load(login\_page)

login\_page.close()

create\_user\_page = QFile("pages/create\_new\_user.ui")

create\_user\_page.open(QFile.ReadOnly)

my\_form5 = loader.load(create\_user\_page)

create\_user\_page.close()

user\_menu\_page = QFile("pages/user\_menu.ui")

user\_menu\_page.open(QFile.ReadOnly)

my\_form6 = loader.load(user\_menu\_page)

user\_menu\_page.close()

chat\_for\_guide\_page = QFile("pages/chat\_page\_for\_guide.ui")

chat\_for\_guide\_page.open(QFile.ReadOnly)

my\_form7 = loader.load(chat\_for\_guide\_page)

chat\_for\_guide\_page.close()

chat\_for\_guide\_page\_heading\_text = "Hello, {name}"

my\_form\_list = [my\_form, my\_form2, my\_form3, my\_form7, my\_form4, my\_form5]

my\_form\_texts\_lists = [my\_form.heading.text(), my\_form2.heading.text(), my\_form3.heading.text(), my\_form7.heading.text()]

sw = QStackedWidget()

for i in my\_form\_list:

sw.addWidget(i)

def toggle\_password(form):

if form.password\_field.echoMode() == QLineEdit.Password:

form.password\_field.setEchoMode(QLineEdit.Normal)

form.toggle\_password.setText("Hide Password")

else:

form.password\_field.setEchoMode(QLineEdit.Password)

form.toggle\_password.setText("Show Password")

def login():

# {"password": "123", "email": "check6@gmail.com"}

mail = my\_form4.email\_field.text()

password = my\_form4.password\_field.text()

data = {"password": password, "email": mail}

print(data)

client.change\_user(None, password, mail)

r = client.send\_login()

print(r)

def create\_new\_user():

name = my\_form5.name\_field.text()

mail = my\_form5.email\_field.text()

password = my\_form5.password\_field.text()

user\_type = my\_form5.user\_type\_input.currentText()

print(name, mail, password, user\_type)

data = {"username": name, "password": password, "email": mail, "type": user\_type}

print(data)

client.change\_user(None, mail, password)

r = client.create\_user(data)

print(r)

def change\_heading\_text(text):

global voice\_client, global\_room\_code

try:

user\_name = client.username

user\_type = client.type

if user\_name is None or user\_type is None:

user\_name = "anon"

user\_type = "normal"

print(user\_name, user\_type)

text = text.replace("{name}", user\_name)

text = text.replace("{user\_type}", user\_type)

except AttributeError as e:

print(e)

except IndexError as e:

print(e)

try:

print(global\_room\_code, "hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh")

text = text.replace("{room\_code}", str(global\_room\_code))

except Exception as e:

print(e)

return text

def change\_window(index):

sw.setCurrentIndex(index)

print(index)

try:

h = my\_form\_texts\_lists[index]

h = change\_heading\_text(h)

my\_form\_list[index].heading.setText(h)

except Exception as e:

print(e)

def enter\_room():

global voice\_client, room\_code

room\_code = my\_form.room\_code\_field.text()

print(room\_code)

change\_window(2)

audio\_client = class\_listener.AudioClient(room\_code, client)

audio\_client.run()

voice\_client = audio\_client

sub\_window = None

def open\_sub\_window():

global sub\_window

if sub\_window is None or not sub\_window.isVisible():

sub\_window = user\_menu.SubWindow(sw)

sub\_window.closed.connect(sub\_window\_closed)

sub\_window.show\_fullscreen()

def sub\_window\_closed():

global sub\_window

sub\_window = None

sub\_chat = None

def open\_sub\_chat():

global sub\_chat, voice\_client

if sub\_chat is None or not sub\_chat.isVisible():

sub\_chat = chat\_menu.ChatSubWindow(sw, voice\_client)

sub\_chat.closed.connect(sub\_chat\_closed)

sub\_chat.show\_fullscreen()

voice\_client.update\_chat\_room(sub\_chat)

def sub\_chat\_closed():

global sub\_chat

sub\_chat = None

def open\_room():

global voice\_client, global\_room\_code

if client.type == "normal":

return

utils.close\_room(client.user\_data)

# Create an AudioRecorder object with the client object and parameters

recorder = class\_broadcast.AudioRecorder(client=client, udp\_ip="127.0.0.1", udp\_port=51166,

host\_listener\_port=51167)

# Start recording in the background

recorder.run()

voice\_client = recorder

global\_room\_code = voice\_client.room\_code

print(global\_room\_code, "ggggggggggggggggggggggggggg")

change\_window(3)

clipboard = app.clipboard()

my\_form.create\_new\_room\_button.clicked.connect(lambda: change\_window(1))

my\_form.connect\_button.clicked.connect(lambda: enter\_room())

my\_form.login\_button.clicked.connect(lambda: change\_window(4))

my\_form2.return\_button.clicked.connect(lambda: change\_window(0))

my\_form2.open\_room\_button.clicked.connect(open\_room)

my\_form3.leave\_room\_button.clicked.connect(lambda: change\_window(0))

my\_form3.open\_chat\_button.clicked.connect(open\_sub\_chat)

my\_form4.password\_field.setEchoMode(QLineEdit.Password)

my\_form4.toggle\_password.clicked.connect(lambda: toggle\_password(my\_form4))

my\_form4.user\_log\_in\_button.clicked.connect(login)

my\_form4.create\_new\_user\_button.clicked.connect(lambda: change\_window(5))

my\_form4.return\_button.clicked.connect(lambda: change\_window(0))

my\_form5.password\_field.setEchoMode(QLineEdit.Password)

my\_form5.user\_type\_input.clear()

for i in ("normal", "guide"):

my\_form5.user\_type\_input.addItem(i)

my\_form5.toggle\_password.clicked.connect(lambda: toggle\_password(my\_form5))

my\_form5.create\_user\_button.clicked.connect(create\_new\_user)

my\_form5.login\_button.clicked.connect(lambda: change\_window(4))

my\_form5.return\_button.clicked.connect(lambda: change\_window(0))

my\_form7.show\_users\_button.clicked.connect(lambda: open\_sub\_window())

my\_form7.open\_chat\_button.clicked.connect(lambda: open\_sub\_chat())

my\_form7.copy\_button.clicked.connect(lambda: clipboard.setText(str(global\_room\_code)))

sw.show()

change\_window(0)

## class\_listener.py

import socket

import pyaudio

import wave

import utils

import random

import threading

import json

class AudioClient:

def \_\_init\_\_(self, room\_code, client=None, chat\_room=None):

if client == None:

self.user\_client = utils.Client(None, None, None)

else:

self.user\_client = client

self.room\_code = room\_code

self.d = utils.get\_host(self.room\_code)

self.UDP\_IP, self.UDP\_PORT, self.host\_listener\_port = self.d["host ip"], self.d["host port"], self.d["host\_listener\_port"]

self.chunk = 1024 # Each chunk will consist of 1024 samples

self.sample\_format = pyaudio.paInt16 # 16 bits per sample

self.channels = 1 # Number of audio channels

self.fs = 44100 # Record at 44100 samples per second

self.time\_in\_seconds = 10

\_, self.PORT, \_ = utils.get\_port\_and\_ip()

self.client = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

self.p = pyaudio.PyAudio()

self.frames = []

self.is\_done = False

self.thread = None

self.chat\_room = chat\_room

def update\_chat\_room(self, chat\_room):

self.chat\_room = chat\_room

def list\_devices(self):

device\_count = self.p.get\_device\_count()

for i in range(0, device\_count):

info = self.p.get\_device\_info\_by\_index(i)

print("Device {} = {}".format(info["index"], info["name"]))

print(self.p.get\_default\_output\_device\_info())

def send(self, msg):

self.client.sendto(msg, (self.UDP\_IP, self.host\_listener\_port))

def send\_d(self, msg):

msg = json.dumps(msg)

self.send(bytes(msg, encoding="utf-8"))

def run(self):

self.thread = threading.Thread(target=self.\_run)

self.thread.start()

def run\_and\_save(self, name):

threading.Thread(target=self.\_run, daemon=True).start()

self.save(name)

def \_run(self):

stream = self.p.open(channels=self.p.get\_default\_output\_device\_info()["maxOutputChannels"],

format=self.sample\_format,

rate=self.fs, # somehow when it's normal it 2 times faster

frames\_per\_buffer=self.chunk,

output=True,

input=False,

output\_device\_index=self.p.get\_default\_output\_device\_info()["index"])

msg = {"msg": "accept me", "name": self.user\_client.username}

self.send\_d(msg)

print("listening")

while not self.is\_done:

data, addr = self.client.recvfrom(1024 \* 100) # buffer size is 1024 bytes

if data == b"stop":

print(data)

self.is\_done = True

if data[0] == 123 and data[-1] == 125:

# Decode the bytes object into a string

string\_data = data.decode('utf-8')

string\_data = string\_data.replace("'", "\"")

print(string\_data)

# Convert the string to JSON

json\_data = json.loads(string\_data)

print(json\_data, type(json\_data))

if json\_data["msg"] == "send msg":

if self.chat\_room is not None:

self.chat\_room.add\_message(f"{json\_data['from']}: {json\_data['content']}")

else:

self.frames.append(data)

stream.write(data)

"""

if random.random() > 0.8:

self.send(b"please activate mic")

"""

stream.stop\_stream()

stream.close()

self.p.terminate()

def save(self, file\_name):

try:

with wave.open(file\_name, 'wb') as file:

file.setnchannels(self.channels)

file.setsampwidth(self.p.get\_sample\_size(self.sample\_format))

file.setframerate(self.fs)

file.writeframes(b''.join(self.frames))

except (IOError, EOFError, ValueError) as e:

print(f"Error occurred while saving the file: {e}")

def connect(self):

self.run()

def send\_msg(self, msg):

msg = {"msg": "send msg", "msg content": msg}

print(msg)

self.send\_d(msg)

if \_\_name\_\_ == "\_\_main\_\_":

audio\_client = AudioClient("nSMQ4PxU")

audio\_client.run\_and\_save("output2.wav")

## class\_broadcast.py

import socket

import pyaudio

import wave

import utils

import threading

import json

import random

class AudioRecorder:

def \_\_init\_\_(self, udp\_ip, udp\_port, host\_listener\_port, client, chat\_room=None):

self.udp\_ip = udp\_ip

self.udp\_port = udp\_port

self.host\_listener\_port = host\_listener\_port

self.client = client

if self.client.type == "normal":

return

self.r = utils.activate\_room(self.client.user\_data, {"host\_ip": self.udp\_ip, "host\_port": self.udp\_port, "host\_listener\_port": self.host\_listener\_port})

self.room\_code = self.r["room code"]

print(self.room\_code)

self.chunk = 1024

self.sample\_format = pyaudio.paInt16

self.channels = 1

self.fs = 44100

self.time\_in\_seconds = 60

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

self.clients = []

self.server = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

self.server.bind((self.udp\_ip, self.host\_listener\_port))

self.p = pyaudio.PyAudio()

self.frames = []

self.chat\_room = chat\_room

def update\_chat\_room(self, chat\_room):

self.chat\_room = chat\_room

def send(self, msg, ip, port):

if self.client.type == "normal":

return

self.sock.sendto(msg, (ip, port))

def broadcast(self, msg):

if self.client.type == "normal":

return

for i in self.clients:

self.send(msg, i[0], i[1])

def broadcast\_msg(self, msg, ip, port):

if self.client.type == "normal":

return

message = msg

print(self.clients)

for i in self.clients:

print("send to: ", i)

msg = {"msg": "send msg", "content": message, "from": self.get\_user\_name(ip, port)}

msg = json.dumps(msg)

self.send(bytes(msg, encoding="utf-8"), i[0], i[1])

def get\_user\_name(self, ip, port):

for item in self.clients:

if item[0] == ip and item[1] == port:

return item[2]

return None

def handle\_connection(self):

if self.client.type == "normal":

return

data, addr = self.server.recvfrom(1024)

print(addr[0], addr[1])

print(data)

if data == b"please activate mic":

msg = {"msg": "listen to", "ip": addr[0], "port": addr[1]}

msg = json.dumps(msg)

self.broadcast(bytes(msg, encoding="utf-8"))

if data[0] == 123 and data[-1] == 125:

print(data)

data = json.loads(data)

if data["msg"] == "accept me":

if addr not in self.clients:

if data["name"] == None:

rand\_nums = [random.randint(1, 100) for i in range(3)]

new\_name = f"anon({''.join(map(lambda x: str(x), rand\_nums))})"

new\_var = random.randint(1, 100)

# Check if the new variable is in the last value of each tuple in the clients list

while any(new\_var in self.clients[-1] for tpl in self.clients):

rand\_nums = [random.randint(1, 100) for i in range(3)]

new\_name = f"anon({''.join(map(lambda x: str(x), rand\_nums))})"

print(new\_name)

name\_to\_add = new\_name

else:

name\_to\_add = data["name"]

client\_to\_add = (addr[0], addr[1], name\_to\_add)

self.clients.append(client\_to\_add)

print(self.clients)

msg = {"msg": "connected"}

msg = json.dumps(msg)

self.send(bytes(msg, encoding="utf-8"), addr[0], addr[1])

elif data["msg"] == "send msg":

print(data["msg content"])

self.chat\_room.add\_message(f"{self.get\_user\_name(addr[0], addr[1])}: {data['msg content']}")

print(self.clients)

self.broadcast\_msg(data["msg content"], addr[0], addr[1])

def list\_devices(self):

if self.client.type == "normal":

return

p2 = pyaudio.PyAudio()

device\_count = p2.get\_device\_count()

for i in range(0, device\_count):

info = p2.get\_device\_info\_by\_index(i)

print("Device {} = {}".format(info["index"], info["name"]))

print(p2.get\_default\_input\_device\_info())

def \_run(self):

if self.client.type == "normal":

return

print('-----Now Recording-----')

stream = self.p.open(format=self.sample\_format,

channels=self.p.get\_default\_input\_device\_info()["maxInputChannels"],

rate=self.fs,

frames\_per\_buffer=self.chunk,

input=True,

output=False,

input\_device\_index=self.p.get\_default\_input\_device\_info()["index"])

print(f"{self.fs=}, {self.chunk=}, {self.time\_in\_seconds=}")

self.frames = []

for i in range(0, int(self.fs / self.chunk \* self.time\_in\_seconds)):

t = threading.Thread(target=self.handle\_connection)

t.daemon = True

t.start()

data = stream.read(self.chunk)

self.frames.append(data)

self.broadcast(data)

stream.stop\_stream()

stream.close()

self.p.terminate()

print('-----Finished Recording-----')

self.broadcast("stop".encode())

print(utils.close\_room(self.client.user\_data))

def run(self):

if self.client.type == "normal":

return

threading.Thread(target=self.\_run, daemon=True).start()

def run\_and\_save(self, name):

if self.client.type == "normal":

return

threading.Thread(target=self.\_run, daemon=True).start()

self.save(name)

def save(self, file\_name):

if self.client.type == "normal":

return

try:

with wave.open(file\_name, 'wb') as file:

file.setnchannels(self.channels)

file.setsampwidth(self.p.get\_sample\_size(self.sample\_format))

file.setframerate(self.fs)

file.writeframes(b''.join(self.frames))

except (IOError, EOFError, ValueError) as e:

print(f"Error occurred while saving the file: {e}")

def send\_msg(self, msg):

msg = {"msg": "send msg", "msg content": msg}

self.broadcast(bytes(str(msg), encoding="utf-8"))

def main():

# Create a client object

client = utils.Client("1", "1", "12345678@gmail.com")

utils.close\_room(client.user\_data)

# Create an AudioRecorder object with the client object and parameters

recorder = AudioRecorder(client=client, udp\_ip="127.0.0.1", udp\_port=51166,

host\_listener\_port=51167)

# Start recording in the background

recorder.run()

if \_\_name\_\_ == "\_\_main\_\_":

main()

## utils.py

import secrets

import socket

import subprocess

import platform

import json

import sys

"""

todo:

"""

def copy\_text(text):

if sys.platform.startswith('win'): # Windows

subprocess.run(['clip'], input=text.strip().encode('utf-16'), check=True)

elif sys.platform.startswith('darwin'): # macOS

subprocess.run(['pbcopy'], input=text.strip().encode('utf-8'), check=True)

elif sys.platform.startswith('linux'): # Linux

try:

subprocess.run(['xclip', '-selection', 'clipboard'], input=text.strip().encode('utf-8'), check=True)

except FileNotFoundError:

try:

subprocess.run(['xsel', '-b'], input=text.strip().encode('utf-8'), check=True)

except FileNotFoundError:

raise Exception("Clipboard not available on this Linux distribution.")

else:

raise Exception("Unsupported platform.")

def get\_ipv4\_address():

try:

system = platform.system()

if system == 'Windows':

output = subprocess.check\_output("ipconfig", shell=True)

elif system == 'Linux' or system == 'Darwin':

output = subprocess.check\_output("ifconfig", shell=True)

else:

return "Unsupported platform"

lines = output.decode("utf-8").split("\n")

for line in lines:

if "inet" in line and system == 'Darwin':

parts = line.split()

return parts[1]

elif "inet" in line and system == 'Linux':

parts = line.split()

if parts[0] == "inet":

return parts[1]

elif "IPv4" in line and system == 'Windows':

parts = line.split(":")

if len(parts) == 2:

return parts[1].strip()

except Exception as e:

print("Error: ", e)

def send\_udp\_large\_data(sock, data, buffer\_size=1024):

# Calculate the number of packets required to send the entire data

packet\_count = len(data) // buffer\_size

if len(data) % buffer\_size != 0:

packet\_count += 1

# Send the data in pieces

for i in range(packet\_count):

# Get the current piece of data to send

start = i \* buffer\_size

end = min(start + buffer\_size, len(data))

data\_chunk = data[start:end]

# Send the current piece of data to the server

sock.sendto(data\_chunk, ("localhost", 5005))

def send\_data(data):

# json the data

data = json.dumps(data)

# Create a socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

try:

# Send the data to the server

send\_udp\_large\_data(sock, bytes(data, encoding="utf-8"))

# Wait for the response

response\_jsond, addr = sock.recvfrom(1024)

except ConnectionResetError:

sock.close()

return {"message": "There is a problem with the server", "code": 104}

# Unjson the response

response = json.loads(response\_jsond)

# Close the socket

sock.close()

return response

def get\_port\_and\_ip():

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.bind(("", 0))

ip\_address, port = sock.getsockname()

sock.close()

sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

sock.bind(("", 0))

ip\_address, port2 = sock.getsockname()

sock.close()

public\_ip = get\_ipv4\_address()

return (public\_ip, port, port2)

ip, port, port2 = get\_port\_and\_ip()

def create\_room\_id():

return secrets.token\_urlsafe(6)

def create\_user(user\_data={"username": "first2", "password": "123", "email": "check6@gmail.com", "type": "guide"}):

# Create the data to send

ip, port, port2 = get\_port\_and\_ip()

data = {"command": "create user", "data": user\_data}

# json the data

data = json.dumps(data)

# Create a socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# Send the data to the server

sock.sendto(bytes(data, encoding="utf-8"), ("localhost", 5005))

try:

# Wait for the response

response\_jsond, addr = sock.recvfrom(1024)

# Unjson the response

response = json.loads(response\_jsond)

except ConnectionResetError:

return {"message": "There is a problem with the server", "code": 104}

# Print the response

print(response["message"])

print("Code: ",response["code"])

# Close the socket

sock.close()

def send\_login(user\_data={"username": "first2", "password": "123", "email": "check6@gmail.com", "type": "guide"}):

# Create the data to send

ip, port, port2 = get\_port\_and\_ip()

data = {"command": "login", "data": user\_data}

# json the data

data = json.dumps(data)

# Create a socket

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

try:

# Send the data to the server

sock.sendto(bytes(data, encoding="utf-8"), ("localhost", 5005))

# Wait for the response

response\_jsond, addr = sock.recvfrom(1024)

except ConnectionResetError:

return {"message": "There is a problem with the server", "code": 104}

# Unjson the response

response = json.loads(response\_jsond)

# Close the socket

sock.close()

return response

def close\_room(user\_data={"username": "first2", "password": "123", "email": "check6@gmail.com"}):

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

data = {"command": "close room", "user\_info": user\_data}

response = send\_data(data)

if response["code"] == 200:

print("Room was closed successfully")

else:

print("Error: " + response["message"])

client\_socket.close()

def activate\_room(user\_data={"username": "first2", "password": "123", "email": "check6@gmail.com"}, data={"host\_ip": ip, "host\_port": port, "host\_listener\_port": port2}):

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

data = {"command": "open room", "user\_info": user\_data, "data": data}

response = send\_data(data)

if response["code"] == 200:

print("Room successfully opened.")

else:

print("Error: " + response["message"])

client\_socket.close()

return response

def get\_host(room\_code=""):

data = {"command": "get host by code", "room\_code": room\_code}

r = send\_data(data)

return r

def get\_user\_type(email):

data = {"command": "get type by email", "email": email}

r = send\_data(data)

if r["code"] == 104:

r["user type"] = None

return r

return r

def get\_user\_name(email):

data = {"command": "get type by name", "email": email}

r = send\_data(data)

if r["code"] == 104:

r["user name"] = None

return r

return r

class BaseClient:

def \_\_init\_\_(self, username=None, password=None, email=None):

if not(password is None or email is None):

self.username = username

self.password = password

self.email = email

self.type = get\_user\_type(self.email)["user type"]

self.user\_data = {"username": self.username, "password": self.password, "email": self.email}

else:

self.username = None

self.password = None

self.email = None

self.type = "normal"

self.user\_data = None

def change\_user(self, username=None, password=None, email=None):

if not(password is None or email is None):

self.username = username

self.password = password

self.email = email

self.type = get\_user\_type(self.email)["user type"]

self.user\_data = {"username": self.username, "password": self.password, "email": self.email}

else:

self.username = None

self.password = None

self.email = None

self.type = "normal"

self.user\_data = None

def activate\_room(self, data={"host\_ip": ip, "host\_port": port}):

if self.user\_data != None:

return activate\_room(self.user\_data, data)

return {"message": "You are not logged in"}

def close\_room(self):

if self.user\_data != None:

return close\_room(user\_data)

return {"message": "You are not logged in"}

def send\_login(self):

if self.user\_data != None:

return send\_login(self.user\_data)

return {"message": "You are not logged in"}

def create\_user(self, data):

if self.user\_data != None:

self.change\_user(data["username"], data["password"], data["email"])

return create\_user(data)

return {"message": "username or password or mail are not valid"}

def get\_host(self, room\_code):

return get\_host(room\_code)

class Client(BaseClient):

def \_\_init\_\_(self, username=None, password=None, email=None):

if not(password is None or email is None):

self.username = get\_user\_name(email)["user name"]

self.password = password

self.email = email

self.type = get\_user\_type(self.email)["user type"]

self.user\_data = {"username": self.username, "password": self.password, "email": self.email}

else:

self.username = None

self.password = None

self.email = None

self.type = "normal"

self.user\_data = None

def change\_user(self, username=None, password=None, email=None):

if not(password is None or email is None):

self.username = get\_user\_name(email)["user name"]

self.password = password

self.email = email

self.type = get\_user\_type(self.email)["user type"]

self.user\_data = {"username": self.username, "password": self.password, "email": self.email}

else:

self.username = None

self.password = None

self.email = None

self.type = "normal"

self.user\_data = None

if \_\_name\_\_ == "\_\_main\_\_":

print(ip, port, port2)

# print(get\_user\_name("23456789@gmail.com"))

# print(get\_port\_and\_ip())

"""

activate\_room({"username": "1", "password": "1", "email": "12345678@gmail.com"}, {"host\_ip": ip, "host\_port": port})

print(get\_host("XzcOnul3"))

printc(create\_room\_id())

print(type(create\_room\_id()))

print(get\_port\_and\_ip())

"""

## Db\_manager.py

import sqlite3

from tkinter import \*

from tkinter import ttk

from tkinter import messagebox

# Establish a connection to the database file

conn = sqlite3.connect('') # name of db file

cursor = conn.cursor()

# Function to fetch data from a table and display it in a tab

def fetch\_table\_data(tab, table\_name):

cursor.execute(f"SELECT \* FROM {table\_name}")

rows = cursor.fetchall()

table\_frame = Frame(tab)

table\_frame.pack(fill='both', expand=True)

# Create a treeview to display the table data

treeview = ttk.Treeview(table\_frame, columns=list(range(len(rows[0]))), show="headings")

treeview.pack(side='left', fill='both', expand=True)

# Set the column headings based on the table columns

cursor.execute(f"PRAGMA table\_info({table\_name})")

columns = cursor.fetchall()

for col in columns:

treeview.heading(col[0], text=col[1])

# Insert the table data into the treeview

for row in rows:

treeview.insert("", "end", values=row)

# Function to add a user to the database

def add\_user(username, password, email, user\_type):

try:

cursor.execute('''INSERT INTO users (username, password, email, type)

VALUES (?, ?, ?, ?)''', (username, password, email, user\_type))

conn.commit()

messagebox.showinfo('Success', 'User added successfully!')

except sqlite3.Error as error:

messagebox.showerror('Error', str(error))

# Function to delete a user from the database

def delete\_user(user\_id):

try:

cursor.execute('DELETE FROM users WHERE id = ?', (user\_id,))

conn.commit()

messagebox.showinfo('Success', 'User deleted successfully!')

except sqlite3.Error as error:

messagebox.showerror('Error', str(error))

# Function to add an active room to the database

def add\_active\_room(room\_code, host\_id, host\_ip=None, host\_port=None, host\_listener\_port=None):

try:

cursor.execute('''INSERT INTO active\_rooms (room\_code, host\_id, host\_ip, host\_port, host\_listener\_port)

VALUES (?, ?, ?, ?, ?)''', (room\_code, host\_id, host\_ip, host\_port, host\_listener\_port))

conn.commit()

messagebox.showinfo('Success', 'Active room added successfully!')

except sqlite3.Error as error:

messagebox.showerror('Error', str(error))

# Function to delete an active room from the database

def delete\_active\_room(room\_code):

try:

cursor.execute('DELETE FROM active\_rooms WHERE room\_code = ?', (room\_code,))

conn.commit()

messagebox.showinfo('Success', 'Active room deleted successfully!')

except sqlite3.Error as error:

messagebox.showerror('Error', str(error))

# Function to view the users and active\_rooms tables

def view\_tables():

cursor.execute('SELECT \* FROM users')

users = cursor.fetchall()

print("Users:")

for user in users:

print(user)

cursor.execute('SELECT \* FROM active\_rooms')

active\_rooms = cursor.fetchall()

print("Active Rooms:")

for room in active\_rooms:

print(room)

# GUI Function to add a user

def add\_user\_gui():

def add\_user\_button():

username = username\_entry.get()

password = password\_entry.get()

email = email\_entry.get()

user\_type = type\_entry.get()

add\_user(username, password, email, user\_type)

add\_user\_window.destroy()

add\_user\_window = Tk()

add\_user\_window.title('Add User')

username\_label = Label(add\_user\_window, text='Username:')

username\_label.pack()

username\_entry = Entry(add\_user\_window)

username\_entry.pack()

password\_label = Label(add\_user\_window, text='Password:')

password\_label.pack()

password\_entry = Entry(add\_user\_window)

password\_entry.pack()

email\_label = Label(add\_user\_window, text='Email:')

email\_label.pack()

email\_entry = Entry(add\_user\_window)

email\_entry.pack()

type\_label = Label(add\_user\_window, text='Type:')

type\_label.pack()

type\_entry = Entry(add\_user\_window)

type\_entry.pack()

add\_button = Button(add\_user\_window, text='Add User', command=add\_user\_button)

add\_button.pack()

add\_user\_window.mainloop()

# GUI Function to delete a user

def delete\_user\_gui():

def delete\_user\_button():

user\_id = user\_id\_entry.get()

delete\_user(user\_id)

delete\_user\_window.destroy()

delete\_user\_window = Tk()

delete\_user\_window.title('Delete User')

user\_id\_label = Label(delete\_user\_window, text='User ID:')

user\_id\_label.pack()

user\_id\_entry = Entry(delete\_user\_window)

user\_id\_entry.pack()

delete\_button = Button(delete\_user\_window, text='Delete User', command=delete\_user\_button)

delete\_button.pack()

delete\_user\_window.mainloop()

# GUI Function to add an active room

def add\_active\_room\_gui():

def add\_active\_room\_button():

room\_code = room\_code\_entry.get()

host\_id = host\_id\_entry.get()

host\_ip = host\_ip\_entry.get()

host\_port = host\_port\_entry.get()

host\_listener\_port = host\_listener\_port\_entry.get()

add\_active\_room(room\_code, host\_id, host\_ip, host\_port, host\_listener\_port)

add\_active\_room\_window.destroy()

add\_active\_room\_window = Tk()

add\_active\_room\_window.title('Add Active Room')

room\_code\_label = Label(add\_active\_room\_window, text='Room Code:')

room\_code\_label.pack()

room\_code\_entry = Entry(add\_active\_room\_window)

room\_code\_entry.pack()

host\_id\_label = Label(add\_active\_room\_window, text='Host ID:')

host\_id\_label.pack()

host\_id\_entry = Entry(add\_active\_room\_window)

host\_id\_entry.pack()

host\_ip\_label = Label(add\_active\_room\_window, text='Host IP:')

host\_ip\_label.pack()

host\_ip\_entry = Entry(add\_active\_room\_window)

host\_ip\_entry.pack()

host\_port\_label = Label(add\_active\_room\_window, text='Host Port:')

host\_port\_label.pack()

host\_port\_entry = Entry(add\_active\_room\_window)

host\_port\_entry.pack()

host\_listener\_port\_label = Label(add\_active\_room\_window, text='Host Listener Port:')

host\_listener\_port\_label.pack()

host\_listener\_port\_entry = Entry(add\_active\_room\_window)

host\_listener\_port\_entry.pack()

add\_button = Button(add\_active\_room\_window, text='Add Active Room', command=add\_active\_room\_button)

add\_button.pack()

add\_active\_room\_window.mainloop()

# GUI Function to delete an active room

def delete\_active\_room\_gui():

def delete\_active\_room\_button():

room\_code = room\_code\_entry.get()

delete\_active\_room(room\_code)

delete\_active\_room\_window.destroy()

delete\_active\_room\_window = Tk()

delete\_active\_room\_window.title('Delete Active Room')

room\_code\_label = Label(delete\_active\_room\_window, text='Room Code:')

room\_code\_label.pack()

room\_code\_entry = Entry(delete\_active\_room\_window)

room\_code\_entry.pack()

delete\_button = Button(delete\_active\_room\_window, text='Delete Active Room', command=delete\_active\_room\_button)

delete\_button.pack()

delete\_active\_room\_window.mainloop()

# GUI Function to view the tables

def view\_tables\_gui():

view\_tables()

messagebox.showinfo('Tables', 'Tables printed in the console.')

# Main GUI window

main\_window = Tk()

main\_window.title('Database Management')

# Create a notebook (tabs) to display tables

notebook = ttk.Notebook(main\_window)

notebook.pack(fill='both', expand=True)

# Get the table names from the database

cursor.execute("SELECT name FROM sqlite\_master WHERE type='table'")

tables = cursor.fetchall()

# Create a tab for each table

for table in tables:

tab = ttk.Frame(notebook)

notebook.add(tab, text=table[0])

fetch\_table\_data(tab, table[0])

# Buttons to perform actions

add\_user\_button = Button(main\_window, text='Add User', command=add\_user\_gui)

add\_user\_button.pack()

delete\_user\_button = Button(main\_window, text='Delete User', command=delete\_user\_gui)

delete\_user\_button.pack()

add\_active\_room\_button = Button(main\_window, text='Add Active Room', command=add\_active\_room\_gui)

add\_active\_room\_button.pack()

delete\_active\_room\_button = Button(main\_window, text='Delete Active Room', command=delete\_active\_room\_gui)

delete\_active\_room\_button.pack()

view\_tables\_button = Button(main\_window, text='View Tables', command=view\_tables\_gui)

view\_tables\_button.pack()

main\_window.mainloop()

# Close the connection to the database

conn.close()

## chat\_page.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">QPushButton {

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="5" column="0">

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="0">

<widget class="QFrame" name="frame">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QGridLayout" name="gridLayout\_2">

<item row="1" column="0">

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="0" column="1">

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="1">

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="2">

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="1">

<widget class="QListView" name="chat\_viewer"/>

</item>

<item row="2" column="1">

<widget class="QPushButton" name="open\_chat\_button">

<property name="text">

<string>Open Chat</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

<item row="2" column="0">

<spacer name="verticalSpacer\_4">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="0">

<widget class="QFrame" name="frame\_2">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QVBoxLayout" name="verticalLayout\_2">

<item>

<widget class="QPushButton" name="pushButton">

<property name="text">

<string>Ask to talk</string>

</property>

</widget>

</item>

<item>

<widget class="QFrame" name="frame\_3">

<property name="enabled">

<bool>true</bool>

</property>

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QPushButton" name="leave\_room\_button">

<property name="text">

<string>Leave room</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="pushButton\_2">

<property name="text">

<string>Change name</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="pushButton\_3">

<property name="text">

<string>Open Chat</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

</layout>

</widget>

</item>

<item row="0" column="0">

<widget class="QLabel" name="heading">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-size:16pt; font-weight:700;&quot;&gt;Hello, {name}&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item row="1" column="0">

<spacer name="verticalSpacer\_5">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

Chat\_page\_for\_guide.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">QPushButton {

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="5" column="0">

<widget class="QFrame" name="frame">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QGridLayout" name="gridLayout\_2">

<item row="0" column="0">

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="0" column="1">

<widget class="QListView" name="listView"/>

</item>

<item row="1" column="1">

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="0" column="2">

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

</item>

<item row="3" column="0">

<widget class="QFrame" name="frame\_2">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QGridLayout" name="gridLayout\_4">

<item row="1" column="0">

<widget class="QPushButton" name="copy\_button">

<property name="text">

<string>copy</string>

</property>

</widget>

</item>

<item row="1" column="1">

<spacer name="horizontalSpacer\_3">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="0" column="0">

<widget class="QLabel" name="heading">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-size:16pt; font-weight:696;&quot;&gt;Hello, {name}, room code: {room\_code}&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

<item row="6" column="0">

<widget class="QFrame" name="frame\_3">

<property name="enabled">

<bool>true</bool>

</property>

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QGridLayout" name="gridLayout\_3">

<item row="0" column="0">

<widget class="QPushButton" name="leave\_room\_button">

<property name="text">

<string>Leave room</string>

</property>

</widget>

</item>

<item row="0" column="1">

<widget class="QPushButton" name="open\_chat\_button">

<property name="text">

<string>Open Chat</string>

</property>

</widget>

</item>

<item row="1" column="0">

<widget class="QPushButton" name="show\_users\_button">

<property name="text">

<string>Show users</string>

</property>

</widget>

</item>

<item row="1" column="1">

<widget class="QPushButton" name="pushButton\_2">

<property name="text">

<string>Close mic/Open mic</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

<item row="7" column="0">

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="0">

<spacer name="verticalSpacer\_4">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

## Create\_new\_user.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">.QPushButton{

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}

.QLabel{

font: 16px &quot;arial&quot;;

}

.QLineEdit{

height: 35%;

font: 24pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="0" column="0">

<widget class="QLabel" name="label\_5">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p align=&quot;center&quot;&gt;&lt;span style=&quot; font-size:36pt; font-weight:700;&quot;&gt;Create New User&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item row="5" column="0">

<widget class="QFrame" name="frame\_6">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QPushButton" name="return\_button">

<property name="text">

<string>Return to Main Page</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="login\_button">

<property name="text">

<string>Log In</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

<item row="3" column="0">

<widget class="QFrame" name="frame\_3">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QVBoxLayout" name="verticalLayout\_3">

<item>

<widget class="QLabel" name="label\_3">

<property name="text">

<string>Password:</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="password\_field"/>

</item>

<item>

<widget class="QRadioButton" name="toggle\_password">

<property name="text">

<string>Show Password</string>

</property>

</widget>

</item>

</layout>

</widget>

</item>

<item row="1" column="0">

<widget class="QFrame" name="frame">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item>

<widget class="QLabel" name="label">

<property name="text">

<string>Name:</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="name\_field"/>

</item>

</layout>

</widget>

</item>

<item row="4" column="0">

<widget class="QFrame" name="frame\_4">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QVBoxLayout" name="verticalLayout\_4">

<item>

<widget class="QLabel" name="label\_4">

<property name="text">

<string>User Type:</string>

</property>

</widget>

</item>

<item>

<widget class="QComboBox" name="user\_type\_input"/>

</item>

<item>

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

</item>

<item row="2" column="0">

<widget class="QFrame" name="frame\_2">

<property name="frameShape">

<enum>QFrame::StyledPanel</enum>

</property>

<property name="frameShadow">

<enum>QFrame::Raised</enum>

</property>

<layout class="QVBoxLayout" name="verticalLayout\_2">

<item>

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Email:</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="email\_field"/>

</item>

</layout>

</widget>

</item>

<item row="6" column="0">

<widget class="QPushButton" name="create\_user\_button">

<property name="text">

<string>Create User</string>

</property>

</widget>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

## Create\_room\_page.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Fixed" vsizetype="Fixed">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">.QPushButton{

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout">

<item row="3" column="0">

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="8" column="0">

<spacer name="verticalSpacer\_4">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="6" column="0">

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QPushButton" name="pushButton">

<property name="text">

<string>Create room and copy code</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="generate\_new\_code\_button">

<property name="text">

<string>Generate new code</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="open\_room\_button">

<property name="text">

<string>Open room</string>

</property>

</widget>

</item>

</layout>

</item>

<item row="4" column="0">

<widget class="QLineEdit" name="lineEdit">

<property name="styleSheet">

<string notr="true">height: 35%;

font: 48pt &quot;MS Shell Dlg 2&quot;;

font: 20pt &quot;MS Shell Dlg 2&quot;;</string>

</property>

</widget>

</item>

<item row="2" column="0">

<widget class="QLabel" name="heading">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-size:14pt; font-weight:696;&quot;&gt;Hello {name}&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item row="0" column="0">

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="5" column="0">

<spacer name="verticalSpacer\_5">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="7" column="0">

<widget class="QPushButton" name="return\_button">

<property name="text">

<string>Return</string>

</property>

</widget>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

## Login\_page.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">.QPushButton{

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}

.QLabel{

font: 16px &quot;arial&quot;;

}

.QLineEdit{

height: 35%;

font: 24pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QGridLayout" name="gridLayout\_2">

<item row="0" column="0">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<widget class="QLabel" name="label\_3">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p align=&quot;center&quot;&gt;&lt;span style=&quot; font-size:36pt; font-weight:700;&quot;&gt;Log In&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item>

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item>

<widget class="QLabel" name="label">

<property name="text">

<string>Email:</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="email\_field"/>

</item>

</layout>

</item>

<item row="1" column="0">

<layout class="QVBoxLayout" name="verticalLayout\_2">

<item>

<widget class="QLabel" name="label\_2">

<property name="text">

<string>Password</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="password\_field"/>

</item>

<item>

<widget class="QRadioButton" name="toggle\_password">

<property name="text">

<string>Show Password</string>

</property>

</widget>

</item>

<item>

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item>

<layout class="QHBoxLayout" name="horizontalLayout">

<item>

<widget class="QPushButton" name="return\_button">

<property name="text">

<string>Return</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="create\_new\_user\_button">

<property name="text">

<string>Create User</string>

</property>

</widget>

</item>

</layout>

</item>

<item>

<widget class="QPushButton" name="user\_log\_in\_button">

<property name="text">

<string>Log In</string>

</property>

</widget>

</item>

</layout>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

## Main\_page.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="enabled">

<bool>true</bool>

</property>

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Fixed" vsizetype="Fixed">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">.QPushButton{

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<layout class="QGridLayout" name="gridLayout">

<item row="5" column="2">

<layout class="QHBoxLayout" name="horizontalLayout\_2">

<item>

<widget class="QPushButton" name="connect\_button">

<property name="text">

<string>Connect</string>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="create\_new\_room\_button">

<property name="styleSheet">

<string notr="true"/>

</property>

<property name="text">

<string>Create new

room</string>

</property>

</widget>

</item>

</layout>

</item>

<item row="0" column="2">

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="2" column="2">

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="7" column="2">

<spacer name="verticalSpacer\_4">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="4" column="2">

<widget class="QLineEdit" name="room\_code\_field">

<property name="styleSheet">

<string notr="true">height: 35%;

font: 48pt &quot;MS Shell Dlg 2&quot;;

font: 20pt &quot;MS Shell Dlg 2&quot;;</string>

</property>

<property name="text">

<string/>

</property>

</widget>

</item>

<item row="1" column="2">

<widget class="QLabel" name="heading">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p align=&quot;center&quot;&gt;&lt;span style=&quot; font-size:28pt; font-weight:600;&quot;&gt;Hello {name}&lt;/span&gt;&lt;/p&gt;&lt;p align=&quot;center&quot;&gt;&lt;span style=&quot; font-size:28pt; font-weight:600;&quot;&gt;You are {user\_type}&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item row="1" column="1">

<spacer name="horizontalSpacer">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="1" column="3">

<spacer name="horizontalSpacer\_2">

<property name="orientation">

<enum>Qt::Horizontal</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>40</width>

<height>20</height>

</size>

</property>

</spacer>

</item>

<item row="3" column="2">

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item row="6" column="2">

<widget class="QPushButton" name="login\_button">

<property name="text">

<string>Log In / Sign In</string>

</property>

</widget>

</item>

</layout>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

## not\_a\_guide\_buy\_license.ui

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>551</width>

<height>728</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<property name="styleSheet">

<string notr="true">.QPushButton{

height:50%;

font: 16pt &quot;MS Shell Dlg 2&quot;;

}</string>

</property>

<widget class="QWidget" name="centralwidget">

<layout class="QVBoxLayout" name="verticalLayout">

<item>

<spacer name="verticalSpacer\_3">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item>

<widget class="QLabel" name="label">

<property name="text">

<string>&lt;html&gt;&lt;head/&gt;&lt;body&gt;&lt;p&gt;&lt;span style=&quot; font-size:28pt; font-weight:700;&quot;&gt;You are not a guide&lt;/span&gt;&lt;/p&gt;&lt;p&gt;&lt;span style=&quot; font-size:28pt; font-weight:700;&quot;&gt;you can buy a guide license&lt;/span&gt;&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>

</property>

</widget>

</item>

<item>

<spacer name="verticalSpacer">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

<item>

<widget class="QPushButton" name="pushButton">

<property name="text">

<string>Buy License</string>

</property>

</widget>

</item>

<item>

<spacer name="verticalSpacer\_2">

<property name="orientation">

<enum>Qt::Vertical</enum>

</property>

<property name="sizeHint" stdset="0">

<size>

<width>20</width>

<height>40</height>

</size>

</property>

</spacer>

</item>

</layout>

</widget>

</widget>

<resources/>

<connections/>

</ui>

## Style.css

QPushButton {

background-color: transparent;

border: 2px solid #3B7080;

border-radius: 5px;

color: #3B7080;

height: 50%;

font: 16pt "Arial 2";

}

QPushButton:hover {

background-color: #3B7080;

color: #FCEC52;

}

QPushButton:hover {

background-color: #3B7080;

color: #FCEC52;

}

QWidget {

background-color: #FCEC52;

color: #3B7080;

}

QMainWindow {

background-color: #FCEC52;

}

QMenuBar {

background-color: #3B7080;

color: #FCEC52;

}

QMenuBar::item {

background-color: transparent;

}

QMenuBar::item:selected {

background-color: #FCEC52;

}

QToolBar {

background-color: #3B7080;

color: #FCEC52;

}

QToolButton {

background-color: transparent;

border: none;

color: #3B7080;

}

QToolButton:hover {

background-color: #FCEC52;

color: #3B7080;

}

QStatusBar {

background-color: #3B7080;

color: #FCEC52;

}

QComboBox {

background-color: transparent;

border: 2px solid #3B7080;

border-radius: 5px;

color: #FFFFFF;

height: 50%;

font: 16pt "Arial 2";

}

QComboBox:hover {

background-color: #3B7080;

color: #FFFFFF;

font: 16pt "Arial 2";

}

## Chat\_menu.py

from PySide6.QtWidgets import QApplication, QMainWindow, QPushButton, QLabel, QVBoxLayout, QHBoxLayout, QWidget, QListWidget, QTextEdit, QLineEdit, QSpacerItem, QSizePolicy

from PySide6.QtGui import QGuiApplication

from PySide6.QtCore import Qt, QObject, Signal, QPropertyAnimation, QPoint, QRect, QEasingCurve, QTimer

from PySide6 import QtCore

class ChatSubWindow(QWidget):

closed = Signal()

def \_\_init\_\_(self, parent=None, client=None):

super().\_\_init\_\_(parent)

if client != None:

self.client = client

else:

self.client = utils.Client(None, None, None)

self.client = client

self.setGeometry(0, 0, 0, 0)

self.setWindowFlag(Qt.FramelessWindowHint)

self.setStyleSheet("background-color: white; border: 1px solid black; color: black;")

# Set up the layout

self.layout = QVBoxLayout()

self.close\_btn = QPushButton("Close")

self.close\_btn.clicked.connect(self.close)

self.layout.addWidget(self.close\_btn)

# Add the chat log

self.chat\_log = QTextEdit()

self.chat\_log.setReadOnly(True)

self.chat\_log.setMinimumHeight(200) # Increase the height of the text box

self.layout.addWidget(self.chat\_log)

# Add the text input and send button

self.input\_layout = QHBoxLayout()

self.input\_text = QLineEdit()

self.input\_text.setFixedHeight(80)

self.input\_layout.addWidget(self.input\_text, 1) # Add stretch to the text input

self.send\_button = QPushButton("Send")

self.send\_button.setFixedHeight(80)

self.send\_button.setFixedWidth(80)

self.send\_button.clicked.connect(self.send\_message)

self.input\_layout.addWidget(self.send\_button)

self.layout.addLayout(self.input\_layout)

# Add some space at the bottom

spacer\_item = QSpacerItem(100, 100, QSizePolicy.Fixed, QSizePolicy.Expanding)

self.layout.addItem(spacer\_item)

self.setLayout(self.layout)

# Create the animation object

self.animation = QPropertyAnimation(self, b"geometry")

self.animation.setDuration(500)

def send\_message(self, msg):

message = self.input\_text.text()

self.input\_text.clear()

if self.client != None:

self.client.send\_msg(message)

def add\_message(self, msg):

self.chat\_log.append(msg)

def show\_fullscreen(self):

desktop\_rect = QGuiApplication.primaryScreen().availableGeometry()

height = desktop\_rect.height() \* 3/4

width = self.parent().width()

self.setGeometry(0, desktop\_rect.height()-height, width, height)

# Set the start and end positions of the animation

start\_pos = QRect(0, desktop\_rect.height(), width, height)

end\_pos = QRect(0, desktop\_rect.height()-height, width, height)

self.animation.setStartValue(start\_pos)

self.animation.setEndValue(end\_pos)

# Start the animation

self.animation.start()

self.show()

def closeEvent(self, event):

self.closed.emit()

super().closeEvent(event)

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

button = QPushButton("Open Sub Window")

button.clicked.connect(self.open\_sub\_window)

self.setCentralWidget(button)

self.sub\_window = None

def open\_sub\_window(self):

if self.sub\_window is None or not self.sub\_window.isVisible():

self.sub\_window = ChatSubWindow(self)

self.sub\_window.closed.connect(self.sub\_window\_closed)

self.sub\_window.show\_fullscreen()

def sub\_window\_closed(self):

self.sub\_window = None

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication([])

window = MainWindow()

window.show()

app.exec()

## user\_menu.py

from PySide6.QtWidgets import QApplication, QMainWindow, QPushButton, QLabel, QVBoxLayout, QHBoxLayout, QWidget, QListWidget, QTextEdit, QLineEdit, QSpacerItem, QSizePolicy

from PySide6.QtGui import QGuiApplication

from PySide6.QtCore import Qt, QObject, Signal, QPropertyAnimation, QPoint, QRect, QEasingCurve, QTimer

from PySide6 import QtCore

class ChatSubWindow(QWidget):

closed = Signal()

def \_\_init\_\_(self, parent=None, client=None):

super().\_\_init\_\_(parent)

if client != None:

self.client = client

else:

self.client = utils.Client(None, None, None)

self.client = client

self.setGeometry(0, 0, 0, 0)

self.setWindowFlag(Qt.FramelessWindowHint)

self.setStyleSheet("background-color: white; border: 1px solid black; color: black;")

# Set up the layout

self.layout = QVBoxLayout()

self.close\_btn = QPushButton("Close")

self.close\_btn.clicked.connect(self.close)

self.layout.addWidget(self.close\_btn)

# Add the chat log

self.chat\_log = QTextEdit()

self.chat\_log.setReadOnly(True)

self.chat\_log.setMinimumHeight(200) # Increase the height of the text box

self.layout.addWidget(self.chat\_log)

# Add the text input and send button

self.input\_layout = QHBoxLayout()

self.input\_text = QLineEdit()

self.input\_text.setFixedHeight(80)

self.input\_layout.addWidget(self.input\_text, 1) # Add stretch to the text input

self.send\_button = QPushButton("Send")

self.send\_button.setFixedHeight(80)

self.send\_button.setFixedWidth(80)

self.send\_button.clicked.connect(self.send\_message)

self.input\_layout.addWidget(self.send\_button)

self.layout.addLayout(self.input\_layout)

# Add some space at the bottom

spacer\_item = QSpacerItem(100, 100, QSizePolicy.Fixed, QSizePolicy.Expanding)

self.layout.addItem(spacer\_item)

self.setLayout(self.layout)

# Create the animation object

self.animation = QPropertyAnimation(self, b"geometry")

self.animation.setDuration(500)

def send\_message(self, msg):

message = self.input\_text.text()

self.input\_text.clear()

if self.client != None:

self.client.send\_msg(message)

def add\_message(self, msg):

self.chat\_log.append(msg)

def show\_fullscreen(self):

desktop\_rect = QGuiApplication.primaryScreen().availableGeometry()

height = desktop\_rect.height() \* 3/4

width = self.parent().width()

self.setGeometry(0, desktop\_rect.height()-height, width, height)

# Set the start and end positions of the animation

start\_pos = QRect(0, desktop\_rect.height(), width, height)

end\_pos = QRect(0, desktop\_rect.height()-height, width, height)

self.animation.setStartValue(start\_pos)

self.animation.setEndValue(end\_pos)

# Start the animation

self.animation.start()

self.show()

def closeEvent(self, event):

self.closed.emit()

super().closeEvent(event)

class MainWindow(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

button = QPushButton("Open Sub Window")

button.clicked.connect(self.open\_sub\_window)

self.setCentralWidget(button)

self.sub\_window = None

def open\_sub\_window(self):

if self.sub\_window is None or not self.sub\_window.isVisible():

self.sub\_window = ChatSubWindow(self)

self.sub\_window.closed.connect(self.sub\_window\_closed)

self.sub\_window.show\_fullscreen()

def sub\_window\_closed(self):

self.sub\_window = None

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication([])

window = MainWindow()

window.show()

app.exec()