# אפליקציה לתקשורת בין אנשים – שליחת הודעות, קבצים וביצוע שיחות - ChatEase

****

בית ספר: ויצו הדסים

מגיש: עומר דגרי

ת.ז: 215200908

שם המנחה: ניר דוויק

תאריך הגשה: 31.05.2023

****

# תוכן עניינים

[אפליקציה לתקשורת בין אנשים – שליחת הודעות, קבצים וביצוע שיחות - ChatEase 1](#_Toc135857972)

[תוכן עניינים 2](#_Toc135857973)

[1. מבוא 3](#_Toc135857974)

[1.1 תכולת ספר הפרויקט 3](#_Toc135857975)

[1.2 הרקע לפרויקט 3](#_Toc135857976)

[1.3 תהליך המחקר 3](#_Toc135857977)

[1.4 אתגרים מרכזיים 3](#_Toc135857978)

[2. מבנה / ארכיטקטורה 4](#_Toc135857979)

[2.1 Front-End 5](#_Toc135857980)

[2.1.1 Login Page 5](#_Toc135857981)

[2.1.2 Signup Page 6](#_Toc135857982)

[2.1.3 Reset Password Page 7](#_Toc135857983)

[2.1.4 האפליקציה עצמה – ChatEase 8](#_Toc135857984)

[2.2 Back-End 9](#_Toc135857985)

[3. מדריך למשתמש 10](#_Toc135857986)

[3.1 צד לקוח 10](#_Toc135857987)

[3.2 צד שרת 11](#_Toc135857988)

[4. בסיס נתונים 12](#_Toc135857989)

[5. מדריך למפתח 14](#_Toc135857990)

[5.1 צד לקוח 14](#_Toc135857991)

[המודול CallsUDP 14](#_Toc135857992)

[המודול ClientSecureSocket 14](#_Toc135857993)

[התיקייה webroot 15](#_Toc135857994)

[מחוץ לתיקיות ולמודולים 15](#_Toc135857995)

[5.2 צד שרת 17](#_Toc135857996)

[המודול CallsUDP 17](#_Toc135857997)

[המודול DirectoryLock 17](#_Toc135857998)

[המודול ServerSecureSocket 17](#_Toc135857999)

[המודול SyncDB 17](#_Toc135858000)

[מחוץ למודולים 18](#_Toc135858001)

[5.3 פרוטוקול התקשורת בין השרת ללקוח 19](#_Toc135858002)

[6. רפלקציה 20](#_Toc135858003)

[7. ביבליוגרפיה 21](#_Toc135858004)

# 1. מבוא

## 1.1 תכולת ספר הפרויקט

ספר זה מתאר את הסיבות להכנת האפליקציה ChatEase, כיצד היא עובדת, תהליך המחקר והפיתוח, מדריך למשתמש ולמפתח.

## 1.2 הרקע לפרויקט

בשנה שעברה עשיתי גם אפליקציית צ’אטים אך ברמה פשוטה יותר ועם GUI שנראה פחות טוב לדעתי, השנה החלטתי לקחת את זה צעד קדימה ולממש עוד פונקציות כמו קבוצות ושיחות, ובנוסף לעשות את זה עם GUI יפה יותר שמאפשר לי לעצב את האפליקציה ברמה הרבה יותר גבוהה (html css js).

## 1.3 תהליך המחקר

פה לא היה הרבה לעשות מאחר ואפליקציות צ’אטים זה משהו שאנחנו משתמשים בחיי היום יום ואנחנו מודעים לאפשרויות שיש בשוק כדון whatsapp, telegram, Instagram, facebook messenger ועוד.  
הפרויקט שלי מאחסן את המידע של הצ’אטים בשרת כך שניתן להתחבר מכל מכשיר חדש למשתמש והמידע ישמר ויהיה את כל ההיסטוריה בלי שזה יתאחסן בחשבון גוגל דרייב שלכם כמו בוואצאפ לדוגמה.  
בנוסף הפרויקט שלי מציע GUI עם כל מיני אפקטים נחמדים בניגוד לשאר האפשרויות בשוק.  
הפרויקט שלי מאפשר לבצע שיחות דרך המחשב דבר שעד לאחרונה לא היה אצל שאר האפליקציות בשוק (למיטב ידיעתי) חוץ מוואצאפ שהוסיפו בשנה האחרונה.

## 1.4 אתגרים מרכזיים

1. חיפוש GUI שיאפשר לי לעצב אותו בצורה שאני אהיה מרוצה ממנה, בסוף בחרתי בספרייה eel בפייתון אשר מאפשר לי לעבוד עם html, css, js ביחד עם הפייתון.  
2. לחשוב על דרך מהירה לנעול את הקבצים של הצאט ברגע שאני משנה אותם בלי לנעול משתמשים שלא נמצאים בצאט או משתמשים שנמצאים בצאט אבל לא מנסים לשלוח הודעה לצאט הזה, על מנת למנוע איבוד הודעות.  
3. כל הקטע של השיחות, הייתי צריך למצוא דרך לעשות את השרת של השיחות מהיר, מאחר ונגיד ומחוברים אליו 5 משתמשים עבור כל חבילה שהוא מקבל (והוא מקבל הרבה) הוא צריך לשלוח 4 חבילות (את אותה החבילה לכל שאר משתתפי השיחה) משמע עבור כל חבילה יש לו פי 4 חבילות לשלוח במקרה הזה אז קרה מצב שאחרי כמה זמן הצטברו אצלי הרבה חבילות בגרסה הראשונה של השרת שיחות מאחר ולא עשיתי אותו מספיק מהיר ואז בתחילת השיחה הכל היה חלק אבל אחרי כמה שניות פתאום התחילו קטיעות (רק בשיחות עם כמה משתמשים).  
4. בהתחלה היו לי בעיות עם הקישור של הפייתון לjs בעזרת הספרייה eel אבל אחרי ששיחקתי עם זה קצת ובדקתי באינטרנט (מהקצת מידע שיש על זה) הצלחתי לעבוד עם זה.

# 2. מבנה / ארכיטקטורה

הפרויקט מחולק ל2 חלקים, חלק ה front end וחלק ה back end, בנוסף חלק הfront end מכיל גם שני חלקים הפייתון וה html css js.

ה front end אחראי על האינטרקציה של המשתמש עם התוכנה וחלק ה back end שאחראי על הטיפול בכל הבקשות של המשתמשים.  
המשתמש לא מתקשר ישירות עם חלק ה back end אלא חלק ה front end מתקשר בשבילו עם ה back end בתגובה למה שהמשתמש עושה.

**משתמש**

בקשת מידע/שליחת מידע

בקשה לשליחת מידע לשרת/בקשת מידע מהשרת

פלט, פקודות

התראות

עדכון הGUI

קבלת מידע

**Server – Back End**

**Python – Front End**

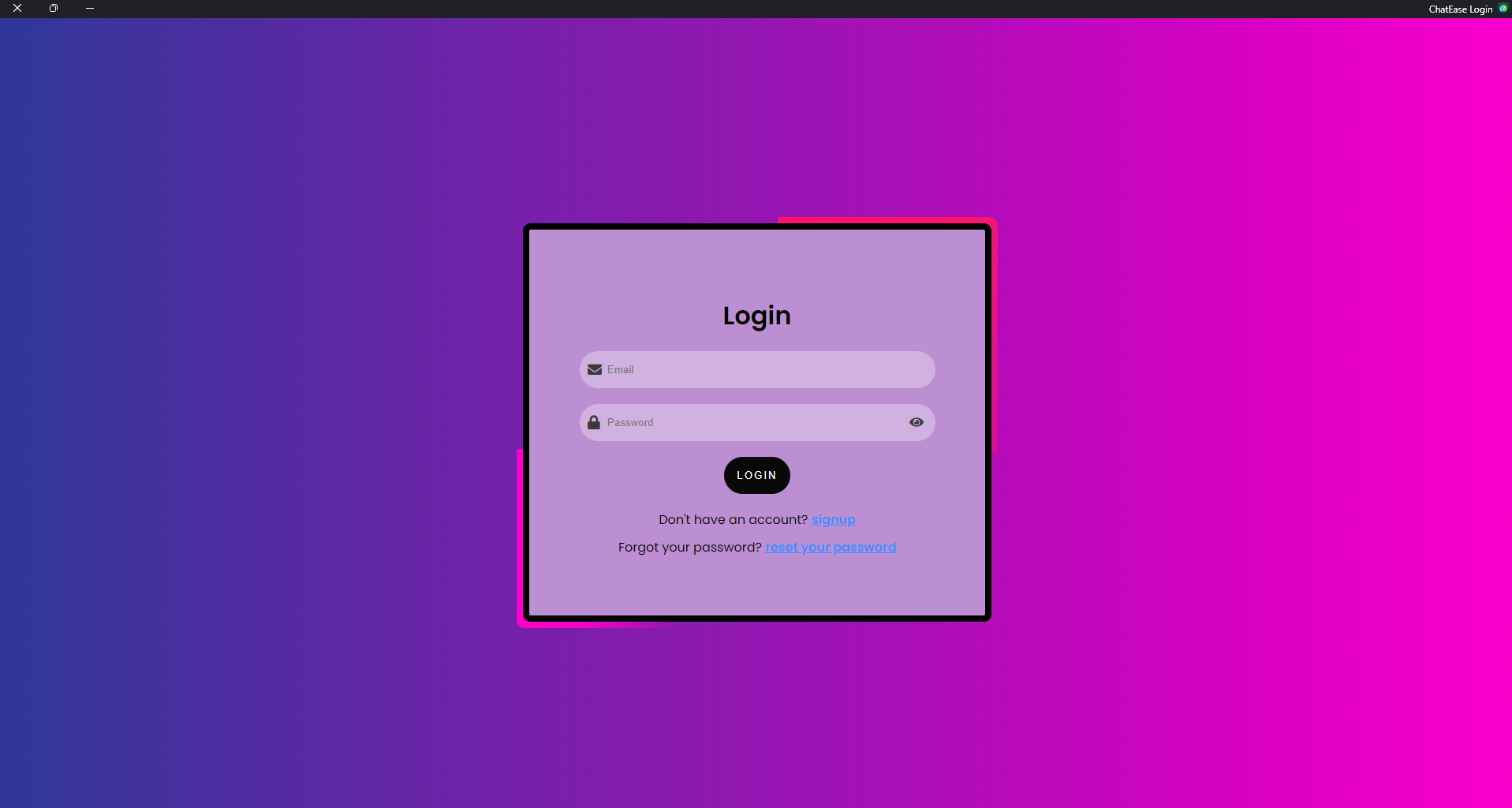
**HTML CSS JS**

## 2.1 Front-End

חלק זה מכיל כמה קבצים שונים בשפות שונות.  
יש את העמודים של הlogin, signup, reset password ואת העמוד של האפליקציה עצמה.

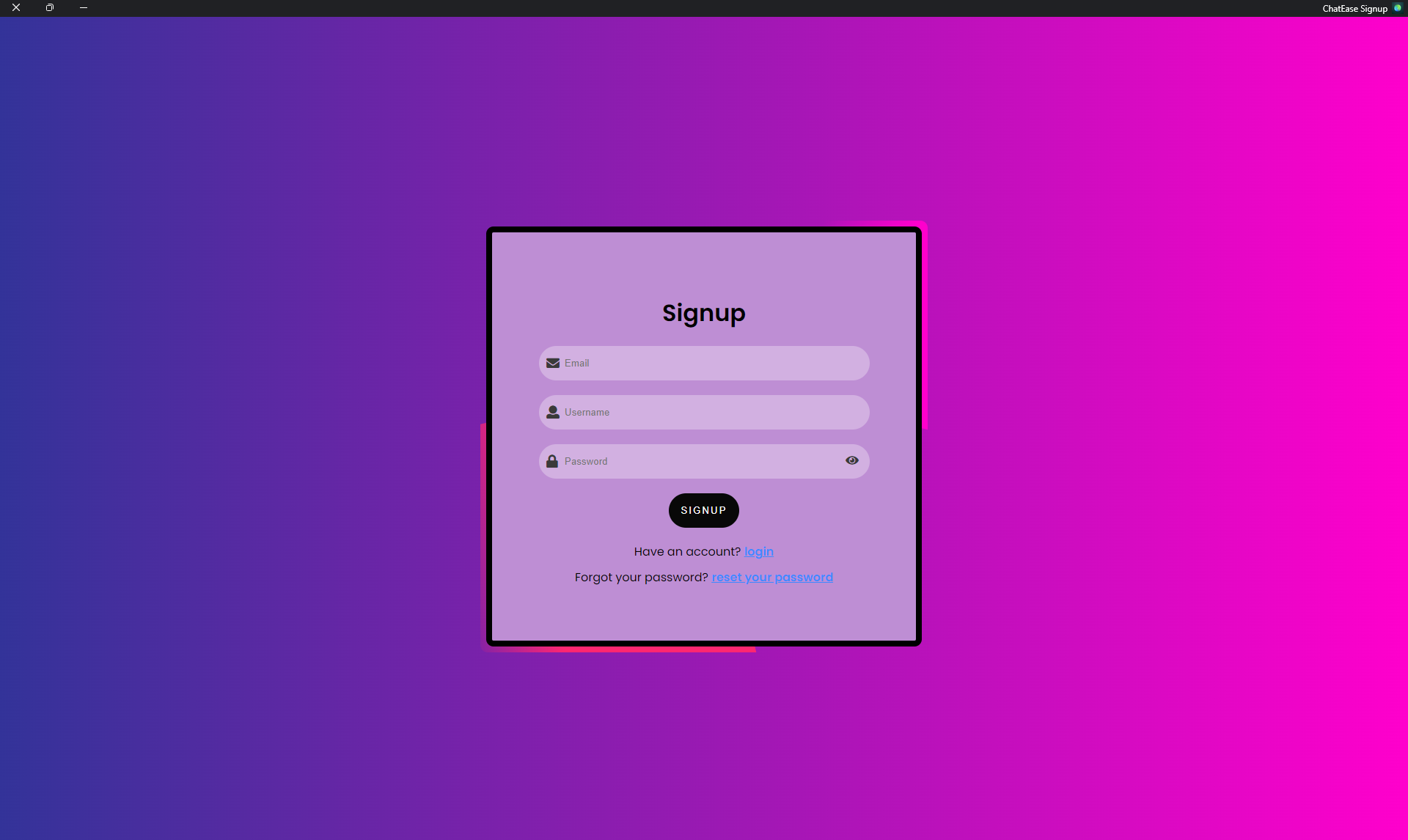
עבור כל אחד מהעמודים האלו יש קובץ html אחד, קובץ css אחד וקובץ js אחד או יותר.  
כל event שקורא שדורש תקשורת עם השרת או קבלת מידע חדש על מנת לעדכן את הGUI הjs מתקשר עם הpython ומקבל את מה שהוא צריך על מנת לבצע את הפעולה. זאת אומרת שעבור כל event כזה יש פונקציה גם בpython שתטפל בה וגם בjs.

### 2.1.1 Login Page

בדף זה ניתן להתחבר למשתמש שקיים או לעבור לאחד מהדפים האחרים - הרשמה או איפוס סיסמה.  
בדף זה המשתמש מתבקש להכניס את האימייל והסיסמה שלו לאפליקציה (ניתן להסתיר או להראות את הסיסמה).  


### 2.1.2 Signup Page

בדף זה ניתן ליצור משתמש חדש או לעבור לאחד מהדפים האחרים - התחברות או איפוס סיסמה.  
בדף זה המשתמש מתבקש להכניס את האימייל, השם משתמש שהוא רוצה ואת הסיסמה שלו לאפליקציה (ניתן להסתיר או להראות את הסיסמה). ולאחר מכן ישלח לאימייל שלו קוד לאישור הרשמה ורק לאחר שיכניס אותו השרת יצור לו משתמש חדש (בתנאי שאין לאימייל משתמש קיים).

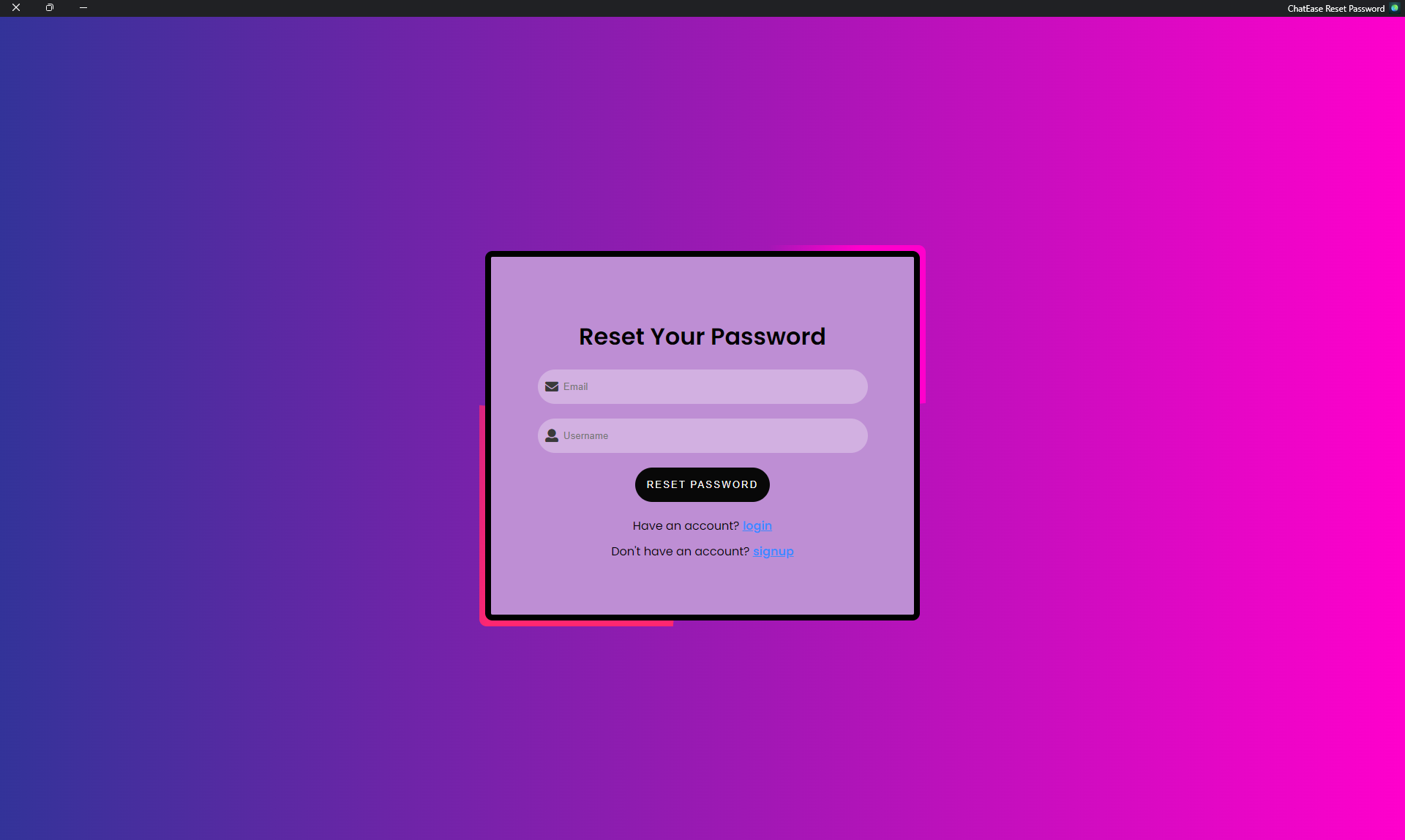


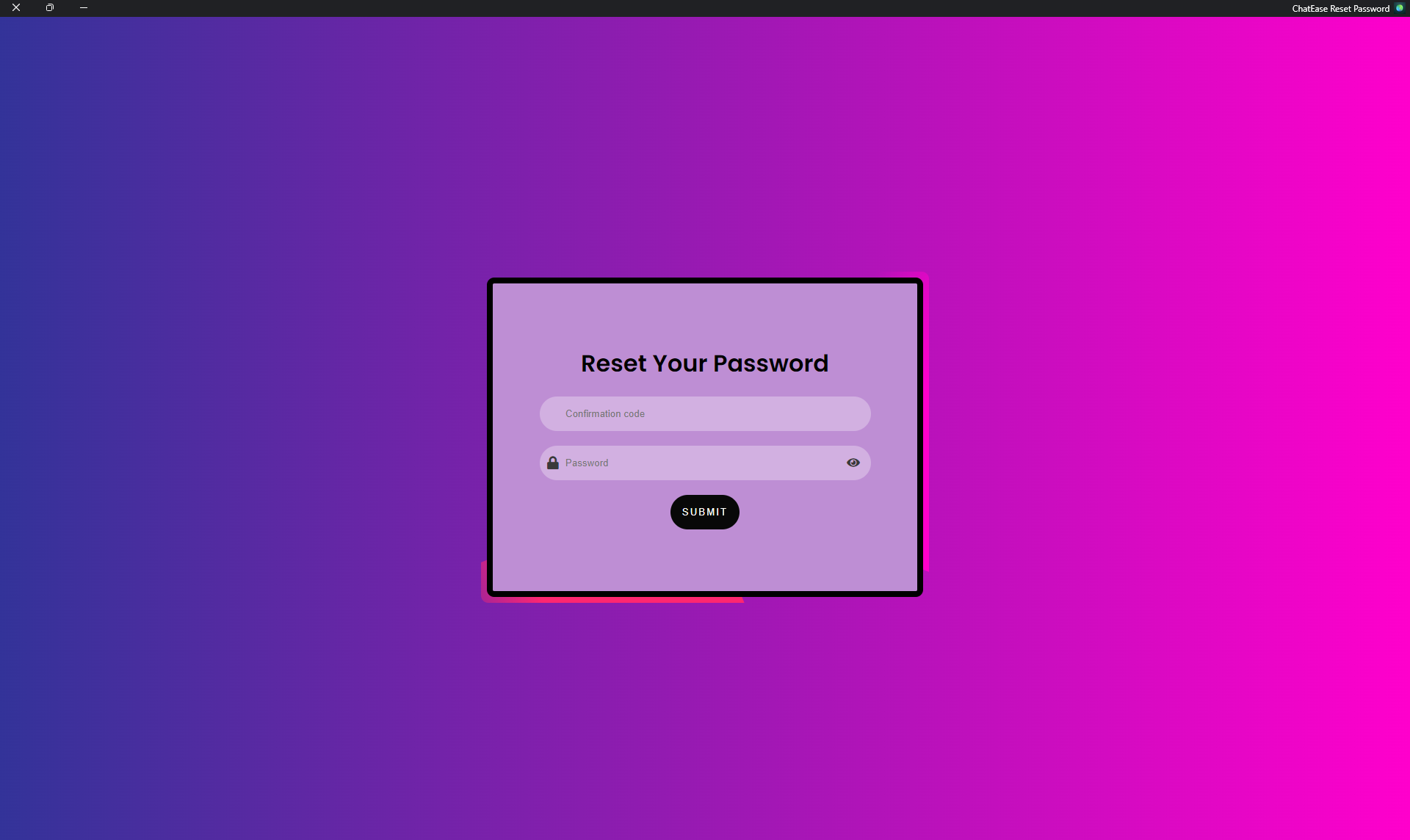
A screenshot of a computer

Description automatically generated with medium confidence

### 2.1.3 Reset Password Page

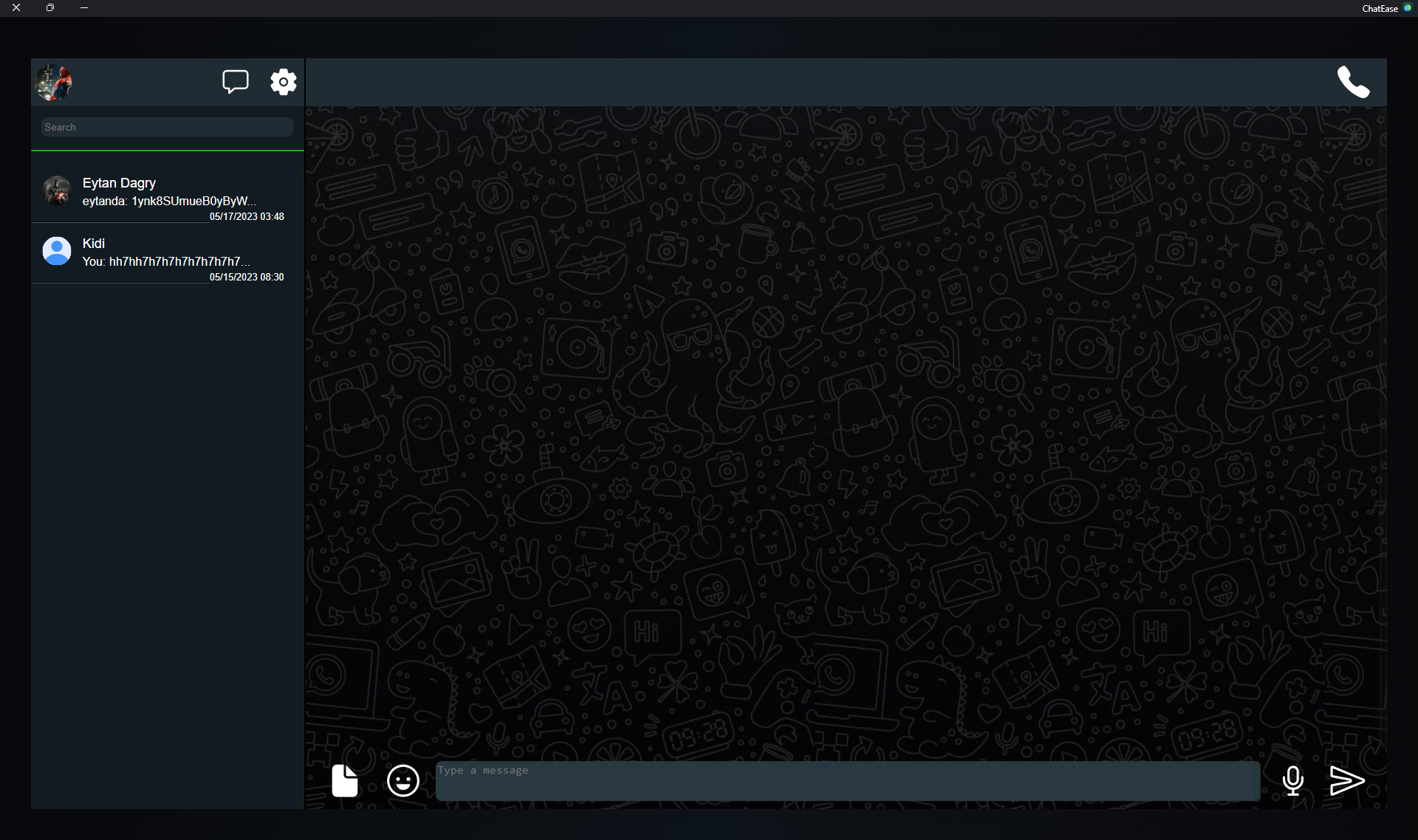
בדף זה ניתן לאפס סיסמה למשתמש קיים או לעבור לאחד מהדפים האחרים - התחברות או הרשמה.  
בדף זה המשתמש מתבקש להכניס את האימייל שלו ואת השם משתמש ולאחר מכן ישלח לו לאימייל קוד לאישור איפוס הסיסמה ויתאפשר לו להכניס את הקוד והסיסמה החדשה אם הקוד נכון תופיע הודעה שהסיסמה אופסה בהצלחה אם לא תופיע הודעת שגיאה.





### 2.1.4 האפליקציה עצמה – ChatEase

בדף הזה המשתמש כבר מחובר ונמצא בתוך האפליקציה, במצב הזה הוא יכול להיכנס לכל הצאטים שלו לשנות תמונת פרופיל להתחיל שיחה לשלוח הודעות וקבצים ללחוץ על הכפתור שליד ההגדרות על מנת ליצור צאטים חדשים ועוד.



A screenshot of a computer

Description automatically generated with medium confidence

## 2.2 Back-End

בצד של השרת יש את הקובץ העיקרי של השרת בנוסף יש 4 מודולים - מודול לשיחות, מודול "לנעילת" קבצים בצורה שלא מפריעה למשתמשים אחרים אלא אם הם מנסים לגשת לקובץ הספציפי הזה בזמן שהוא נעול, מודול לסוקט עבור הצד של השרת שהוא מאובטח ובעל פרוטוקול (שליחת אורך הודעה מאופס ל30 תווים ואז את ההודעה) ומודול לdatabase אשר מסונכרן בין threads ובין processes שזה היה עבודה שעשינו השנה בלימודים והחלטתי להשתמש בה לפרויקט (שיניתי כמה דברים).

השרת מתחיל מיצירת כל הdatabaseים הנחוצים – אימייל סיסמה, אימייל שם משתמש, id של צאט וכל המשתמשים שבו, אימייל סטטוס (מחובר או לא ואם לא מתי היה לאחרונה). כל אלו משתמשים במודול של הdatabase.

לאחר מכן הוא מתחיל מלמחוק את כל המנעולים של המודול לנעילת קבצים במקרה והשרת נעצר על ידי KeyboardInterrupt ולא סיים כמו שצריך.

רק אחרי כל זה מתחילה הפונקציה main בפונקציה זו השרת מייצר לו מפתחות public וprivate לאחר מכן יוצר סוקט עבור השרת עם המודול ServerSecureSocket (המודול שאזכרתי למעלה) אחרי זה הוא שולח מייל עם הip החיצוני שלו אל מייל משותף שיצרתי למשתמשים (למשתמשים אין גישה ישירה רק לפייתון), עשיתי את זה בעיקר בגלל שכשבדקתי את האפליקציה עם חברים לא היה לי כוח כל הזמן להעביר להם את הip החיצוני שלי ושהם יתחילו לשנות בקוד וגם שיצרתי exe לא היה ניתן לשנות את הקוד אז זה היה יותר בשביל שהפייתון יוכל לבדוק לבד מה הip של השרת כרגע, אני מודע שזה לא הדבר הכי חכם כי זה אימייל שנמצא אצל כל המשתמשים אבל זה היה יותר בשביל השלב של הבדיקה ולא באמת בתור מטרה לשימוש כפיצ’ר.

בשלב זה השרת מחכה ללקוחות חדשים ועבור כל לקוח הוא פותח thread חדש שדואג לטפל בו וכך גם מאפשר לעטוף כל לקוח במקרה ויש איזה exception הוא לא ישפיע על האחרים.

Client Front-End

Response with or without data

Request with or without data

Return the status and some information if there is

Transfer request to the needed function

**handle\_client**

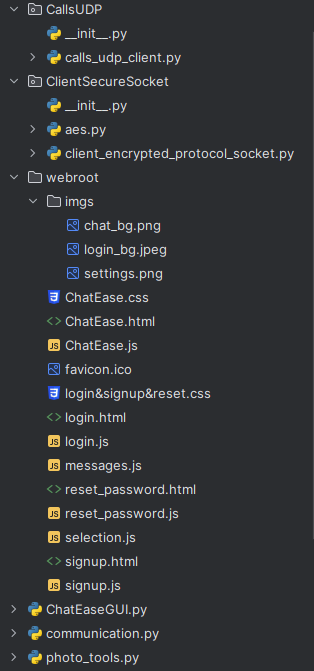
Some function to handle this specific request

# 3. מדריך למשתמש

## 3.1 צד לקוח

אני יוצר exe של הצד של הלקוח אז כל מה שצריך זה לפתוח את הexe ליצור משתמש / לאפס סיסמה ואו להתחבר ולאחר מכן יש גישה לאפליקציה עצמה. כשמריצים את הexe זה כנראה יזהיר שזה יכול להיות ווירוס מאחר ואין לי דרך לחתום על הexe בתור חברה כי אין לי חברה, צריך ללחוץ על המשך בכל מקרה ורק אז האפליקציה תתחיל.  
לדפים השונים של התחברות, הרשמה, איפוס סיסמה והאפליקציה עצמה ניתן לראות [2.1 Front-End](#_2.1_Front-End).

במקרה של שימוש ישירות בפייתון יש צורך בכל הקבצים הבאים כפי שהם מסודרים בתיקיות:



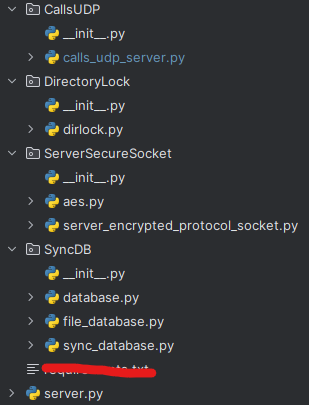
בנוסף בפייתון צריך את הספריות הבאות:  
eel  
numpy  
pillow  
pyaudio  
easygui  
pycryptodome

והגרסת פייתון צריכה להיות 3.11 ומעלה בעקבות הצורה של התיעוד של הtypes בתוך הקבצי פייתון המאופשרת רק מגרסה זו ומעלה (ניתן גם בגרסאות קודמות אם מוסיפים בראש הקובץ from \_\_future\_\_ import annotations), יכול להיות שניתן גם בגרסה 3.10 אך אני לא בטוח.

## 3.2 צד שרת

גם פה, אני יוצר exe כך שאין צורך לעשות משהו מיוחד חוץ מלהריץ אותו.  
לפירוט על התהליך של השרת ניתן לראות [2.2 Back-End](#_2.2_Back-End).

במקרה שמריצים ישירות את הפייתון יש צורך בכל הקבצים הבאים כפי שהם מסודרים בתיקיות:



בנוסף בפייתון צריך את הספריות הבאות:  
rsa  
pycryptodome

והגרסת פייתון צריכה להיות 3.11 ומעלה בעקבות הצורה של התיעוד של הtypes בתוך הקבצי פייתון המאופשרת רק מגרסה זו ומעלה (ניתן גם בגרסאות קודמות אם מוסיפים בראש הקובץ from \_\_future\_\_ import annotations), יכול להיות שניתן גם בגרסה 3.10 אך אני לא בטוח.

# 4. בסיס נתונים

הבסיסי נתונים נמצאים בצד של השרת וכאשר לקוח מתחבר מועבר אליו כל המידע שלו וכל הצ’אטים שלו.

מידע עבור השרת על המשתמשים:  
A screenshot of a computer

Description automatically generated with medium confidence

אלו כל הdatabaseים של השרת (בסופו של דבר זה dict אך זה מסונכרן בין threads):  
1. הid של כל הצ’אטים ועבור כל id כל המשתמשים בצ’אט. מפתח str ערך list[str]  
2. אימייל של כל משתמש והסיסמה שלו (עוברת hash פעמיים, פעם אחת ישר אחרי שהמשתמש מכניס את הסיסמה בצד של הלקוח ועוד אחד ישר שהיא מגיעה לשרת כדי לא לשמור ישירות את הhash שאיתו ניתן להתחבר). מפתח str ערך str  
3. אימייל של כל משתמש והשם משתמש שלו (כדי לאפשר לכמה אימייל להיות עם אותו של משתמש). מפתח str ערך str  
4. האימייל של כל משתמש והסטטוס שלו (מחובר וכמה סוקטים שלו מחוברים או לא מחובר ומתי היה מחובר לאחרונה). מפתח str ערך list[str, datetime.datetime | int]

בנוסף המידע של כל הצ’אטים והמשתמשים נשמר בקבצים כפי שמתואר להלן:  
A screenshot of a computer

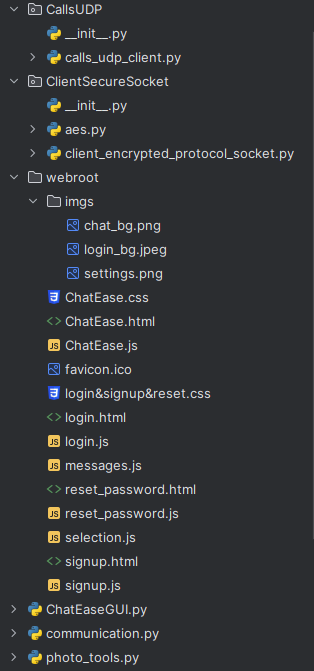
Description automatically generated

**בצ’אטים:** יש קובץ שמכיל את שם הצ’אט, קובץ שמכיל את סוג הצ’אט (יחיד או קבוצה) קובץ שמכיל את מספר ההודעות שלא נראו על ידי כל משתמש וקובץ שמכיל את כל המשתמשים שנמצאים בצ’אט.  
בנוסף יש את התיקייה שנקראת data ובתוכה את התיקיות chat וfiles בתיקייה של files פשוט יש את כל הקבצים שנשלחו בצ’אט, בתיקייה של הchat יש x קבצים (עבור כל 800 הודעות קובץ) כל קובץ כזה מכיל מילון שעבר serialization ו"נזרק" לקובץ המילון מכיל את הindex של ההודעה, את ההודעה עצמה וכל מיני meta data על ההודעה כגון מי ראה מי מחק עבור עצמה האם נמחקה לכולם (במקרה כזה ההודעה באמת נמחקת גם) הזמן שנשלחה ועוד.

**במשתמשים:** יש קובץ שמכיל את כל הצ’אטים שהם נמצאים בהם (chat id), קובץ של התמונת פרופיל שלהם, קובץ של האימיילים של כל המשתמשים שהם מכירים, קובץ של כל הצ’אטים שהם נמצאים בהם שהם לא קבוצות אלא אחד על אחד, הקובץ של הnew\_data שמופיע בתמונה כבר לא קיים.

# 5. מדריך למפתח

## 5.1 צד לקוח



המודול CallsUDP

calls\_udp\_client.py – מכיל 3 פונקציות עיקריות ועוד פונקציה אחת שמתחילה את כולן, 2 פונקציות לשליחה וקבלה של חבילות UDP של אודיו ועוד פונקציה לשמירה על קשר TCP בנוסף המאפשר הזדהות וזיהוי בקלות יותר אם הלקוח עדיין מחובר (ניתן גם בלי).

המודול ClientSecureSocket  
1. :aes.py  
הקובץ מכיל מחלקה בשם AESCipher אשר משומשת על מנת להצפין ולפענח את כל התקשורת בין הלקוחות לשרת. במחלקה יש 4 פונקציות, ורק ל2 מהם צריך באמת לקרוא – encrypt, decrypt אשר מצפינות ומפענחות את המידע באמצעות הפרוטוקול AES השתיים האחרות הן \_pad, \_unpad אשר משומשות על מנת להשלים את האורך של המידע שיהיה באורך שמתאים להצפנה ובשביל להוריד את המידע המיותר שנוסף כדי להצפין.  
*2. Client\_encrypted\_protocol\_socket.py:*  
הקובץ מכיל מחלקה בשם ClientEncryptedProtocolSocket אשר מכילה 7 פונקציות לשימוש חיצוני – recv\_message לקבלת הודעה 1 מהשרת, send\_message לשליחת הודעה לשרת, connect בשביל להתחבר לשרת, set\_timeout בשביל לשנות את הtimeout של הסוקט, get\_timeout כדי לקבל את הtimeout שיש כרגע, getpeername כדי לקבל את הip והport שאליו אנחנו מחוברים, close על מנת לסגור את הסוקט.  
ו2 פונקציות לשימוש פנימי - \_\_recv\_all על מנת לוודא קבלה של כל המידע, \_\_exchange\_aes\_key אשר מבצעת החלפה עם השרת של המפתח הציבורי שהוא מייצר בהתחלה על מנת לשלוח לו בצורה מוצפנת את המפתח להצפנת AES שתשמש אותנו להמשך התקשורת מאחר והצפנה זו מהירה יותר על מידע גדול באופן משמעותי.

התיקייה webroot  
*1. התיקייה imgs* - מכילה 3 תמונות עבור הGUI  
*2. ChatEase.css* - הקובץ עיצוב של האפליקציה עצמה  
*3. ChatEase.html* - הקובץ הבסיסי לאפליקציה עצמה, לא מכיל יותר מידי רק את השלד של האתר כל השאר נעשה מהjs.  
*4. ChatEase.js* - אשר מכיל את כל הפונקציות לטיפול בלחיצה על כל כפתור וכל הפונקציות לתקשורת עם הpython על מנת לקבל את כל הצ’אטים והמשתמשים המוכרים והתמונות וכו’.

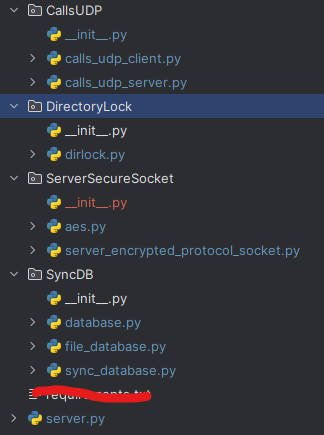
*5. favicon.ico* - הicon של האתר  
*6. Login&signup&reset.css* – הקובץ עיצוב של העמודים של ההתחברות, הרשמה ואיפוס סיסמה.  
*7. login.html – מכיל את הדף התחברות*8. login.js *– מכיל כמה פונקציות בודדות אשר מעבירות לpython את השם משתמש והסיסמה כדי שיוכל להזדהות מול השרת.*9. messages.js *– מכיל את כל סוגי ההודעות על מנת שיהיה אפשר לשכפל אותן כל פעם שיוצרים הודעה במקום ליצור מחדש כל פעם (גם נראה נקי יותר בקוד וגם אמור לעבוד מהר יותר השכפול מאשר היצירה).*10. reset\_password.html *– מכיל את הדף לאיפוס סיסמה.*11. reset\_password.js *– מכיל כמה פונקציות בודדות לתקשורת עם הpython.*12. Selection.js *– לא קיים יותר. היה בהתחלה כדי לא לאפשר לבחור טקסט ואז גיליתי שאפשר לעשות את זה מהcss.*13. signup.html *– מכיל את הדף להרשמה.*14. signup.js *– מכיל כמה פונקציות בודדות לתקשורת עם הpython.*

מחוץ לתיקיות ולמודולים  
*1. communication.py* – מכיל פונקציות להרשמה ואיפוס סיסמה, ומכיל מחלקה בשם Communication אשר משמשת לביצוע כל התקשורת עם השרת ברגע שהצד של הלקוח יודע מה השם משתמש והסיסמה של הלקוח, במחלקה זו ישנם 17 פונקציות:  
1.1. login – להתחברות רגילה.  
1.2. login\_sync – להתחברות של סוקט המיודעת לסנכרון.  
1.3. sync – מחכה להודעת סנכרון מהשרת (כל פעם שיש מידע חדש) ושומרת את המידע החדש לקבצים.  
1.4. upload\_file – פותחת thread לupload\_file\_ ומתחילה אותו.  
1.5. upload\_file\_ - מעלה קובץ לשרת (בתור הודעה לצאט מסוים), לא אמורים לקרוא לה, רק לupload\_file.  
1.6. send\_message – בשביל לשלוח הודעה בצאט  
1.7. familiarize\_user\_with – בשביל לבדוק האם קיים משתמש ואם כן "להכיר" בינו לבין המשתמש הזה, בשביל לאפשר להתחיל איתו צאט.  
1.8. new\_chat – על מנת ליצור צאט חדש (אחד על אחד).  
1.9. new\_group – על מנת ליצור קבוצה חדשה.  
1.10. add\_user\_to\_group – על מנת להוסיף משתמש לקבוצה מסוימת.  
1.11. remove\_user\_to\_group – על מנת להסיר משתמש מסוים מקבוצה מסוימת.  
1.12. make\_call – על מנת להתחיל שיחה עם צאט מסוים (שיחה קולית).  
1.13. upload\_profile\_picture – על מנת להעלות תמונת פרופיל חדשה.  
1.14. upload\_group\_picture – על מנת לשנות את התמונה של קבוצה מסוימת.  
1.15. delete\_message\_for\_me – על מנת למחוק הודעה מסוימת עבורי.  
1.16. delete\_message\_for\_everyone – על מנת למחוק הודעה מסוימת עבור כולם.  
1.17. mark\_as\_seen – על מנת להודיע לשרת באיזה צאט המשתמש נמצא כדי שיוכל לסמן שכל ההודעות *בצאט הזה נקראו על ידי המשתמש הזה.*

2. ChatEaseGUI.py *– מכיל בעיקר פונקציות שעוטפות את כל הפונקציות בcommunication כדי לחשוף אותן לjs באמצעות eel. בנוסף מכיל כמה פונקציות שחשופות לjs כדי לקבל מידע על צאט/משתמשים מוכרים ועוד פונקציה שנמצאת בלולאה של סנכרון.*

3. photo\_tools.py *– עבור בדיקת גודל תמונה (שיהיה לפחות 64x64) יש גם פונקציה שהופכת תמונה לעגולה אך היא לא בשימוש יותר.*  
**\*\*\*\* אני רוצה לציין שגם כל הפונקציות וגם המשתנים בקוד מתועדים אז לא נכנסתי ממש לעומק כאן אלא נתתי פירוט כללי של איזה פונקציות קיימות והשמות שלהן ברורים \*\*\*\***

## 5.2 צד שרת



### המודול CallsUDP

Calls\_udp\_server.py – גם פה מכיל 3 פונקציות עיקריות ועוד 2, אחת שעוטפת את זאת שמתחילה את 3 הפונקציות העיקריות ואחת שמתחילה את שלושת הפונקציות העיקריות. הפונקציות העיקריות – קבלה של חבילות מכל הלקוחות, הפצה של כל חבילה לכל הלקוחות חוץ מזה ששלך את החבילה, פונקציה לשמירה על חיבור TCP בנוסף לUDP על מנת הזדהות וזיהוי קל יותר של ניתוק.

### המודול DirectoryLock

מכיל רק קובץ אחד אשר מכיל 2 פונקציות – block, unblock אשר יוצר תיקייה על מנת לחסום משאב ומסיר את התיקייה על מנת לפנות את המשאב, מה שיקרה בעצם אם 2 ינסו לעשות block זה שאחד מהם יקבל exception שהתיקייה קיימת כבר וכך ידע שהמשאב חסום.

### המודול ServerSecureSocket

1. aes.py – כמו בלקוח [5.1 צד לקוח](#_5.1_צד_לקוח).
2. server\_encrypted\_protocol\_socket.py – גם כמו בלקוח רק הexchange של המפתח AESשונה ויש פונקציה של accept ואין של connect וכו’ הבדלים כרגיל בין סוקט של שרת ושל לקוח.

### המודול SyncDB

1. database.py – מכיל מחלקה אחת, הכי בסיסית, בסך הכל עוטפת את המילון של python.
2. file\_database.py – מכיל מחלקה אחת אשר יורשת מהמחלקה בdatabse.py ומוסיפה כתיבה לקובץ וקריאה מקובץ על מנת לאפשר סנכרון בין threads ובין processes.
3. sync\_database.py – מכיל מחלקה אשר מקבלת עצם של FileDatabase אשר נמצאת בקובץ file\_database.py ומאפשרת set וget כמו על dictionary זאת אומרת בעזרת [] אך שימו לב מאחר וזה משותף בין threads ובין processes אם משנים ערך בתוך המילון צריך לעשות עליו set ולא משהו כמו:  
   database[“5”].append(5)   
   מאחר וזה לא יגרום לטריגר של כתיבה לקובץ, במקום זאת צריך לעשות:   
   some\_list = database[“5”] ואז צריך   
   some\_list.append(5) ואז   
   database[“5”] = some\_list

### מחוץ למודולים

server.py – מכיל את כל הפונקציות לטיפול בכל הבקשות של הלקוח ופונקציה לטיפול בconnection עם הלקוח – handle\_client, עוד פונקציה לקבלת לקוחות ובשביל להתחיל את השרת – main ועוד כמה פונקציות לשימוש כללי של השרת. אני לא מפרט פה מאחר וכל הפונקציות מתועדות גם עם תיעוד במילים של מה הן עושות וגם type hints, כך שאין צורך לפרט עוד פעם.

**\*\*\*\* אני רוצה לציין שגם כל הפונקציות וגם המשתנים בקוד מתועדים אז לא נכנסתי ממש לעומק כאן אלא נתתי פירוט כללי של איזה פונקציות קיימות והשמות שלהן ברורים ומתארים את מה שהן עושות \*\*\*\***

## 5.3 פרוטוקול התקשורת בין השרת ללקוח

הפרוטוקול לא מסובך, הפרוטוקול להלן:

cmd (padded to length of 30 chars) after the cmd without any separation if there is only one data argument then it will be straight after the cmd if there are multiple data arguments we will put the length of the argument (padded to 15 chars) and then the argument data, the last argument won’t have length because it will just be the rest of the message.  
after we create a request/response in this format we will send it with the special ServerSecureSocket module or ClientSecureSocket, which will send the length of the request/response (padded to 30 chars) and only then the request/response itself, when receiving we receive 30 chars that will tell us the length of the request/response and then the data itself.

יש גם הודעה של סנכרון שבמקרה הזה פשוט מעבירים מילונית דרך הסוקט (בעזרת pickle, אשר גם נעטפת ב30 תווים שאומרים מה האורך של כל ההודעה) והמילונית מכילה בתור מפתח path של קובץ ובתור ערך את הdata של הקובץ.

# 6. רפלקציה

**איך הייתה עבורו העבודה על הפרויקט?**

היה לי כיף לעבוד על הפרויקט, חשבתי על זה עוד משנה שעברה אחרי שעשיתי את הפרויקט שאני רוצה לעשות השנה גרסה טובה יותר של הפרויקט של יא עם יותר פיצ'רים ושתראה גם הרבה יותר טוב, שנה שעברה השתמשתי בtkinter ואפשר לעשות שם דברים יפים אבל זה לא מתקרב בכלל למה שאפשר לעשות עם html ועם css. בהתחלה השנה השתמשתי גם עם tkinter וחיפשתי דרכים לעשות את זה יפה יותר ושראיתי שאין כלכך דרך ממש לעצב את כל הwidgets החלטתי לחפש משהו אחר.

**מה הקשיים / אתגרים שעמדו בפניו**

בהתחלה עשיתי את הפרויקט עם סוקט של ssl ויש אצלהם בעיה שמעבירים הרבה מידע בפעם אחת זה פשוט נתקע אז הייתי צריך לממש את זה בעצמי, זה החלק של הaes.py ושל הclient\_encrypted\_protocol\_socket.py ושל הsever\_encrypted\_protocol\_socket.py, אשר מחליפים מפתח חיצוני ומעבירים את המפתח לAES להמשך התקשורת בצורה מוצפנת (עם RSA עם המפתח הציבורי של השרת) ואז כל התקשורת ממשיכה עם הצפנת AES, למדתי ככה עוד קצת על הצפנה.

בנוסף כל הקטע של השיחות זה היה משהו שלא עשיתי אף פעם וידעתי שזה לא יהיה הכי פשוט אבל כשהתחלתי ממש את השיחות ראיתי שצריך למצוא דרך לעשות את השרת ממש מהיר ויעיל אחרת יהיה תקיעות. בגרסה הראשונה של השיחות שעשיתי השרת לא היה מספיק מהיר (עבור כמה משתמשים) וראיתי שמה שקורה זה שבכמה שניות הראשונות זה עובד חלק ואחרי זה מתחיל להצטבר חבילות אצל השרת שהוא לא מספיק להפיץ לכל הלקוחות ואז מתחיל להיתקע.

**מה הוא היה עושה אחרת לו היה מתחיל היום**

אם הייתי מתחיל את הפרויקט מחדש היום עם כל הידע הייתי הולך ישר על html ועל css וגם לא הייתי משתמש בeel כדי לאפשר לגשת לזה דרך דפדפן בלי python אך בגלל שכשעברתי לhtml ולcss וjs השרת כבר היה מוכן והוא לא עובד בhttp אלא בפרוטוקול שאני עשיתי חיפשתי דרך לעבוד עם דרך הpython עם html css js כדי שלא אצטרך לשנות את כל השרת כי לא ידעתי אם יהיה לי זמן גם לזה.

**מה אם היה קורה אחרת העבודה הייתה יעילה יותר עבורו**

עשיתי גם ניתוח להסרת משקפיים לקראת הסוף של העבודה על הפרויקט וזה השבית אותי לשבוע שהיה קריטי אבל הסתדרתי.  
בנוסף אם הייתי מחליט מראש ללכת על html css זה היה חוסך לי הרבה עבודה כי החלפתי לזה רק אחרי שכבר סיימתי את הGUI של המשתמשים וזה היה הרבה עבודה שבסוף הלכה לפח כי עברתי מtkinter לhtml וcss.

**מה אתם מרגישים שהעבודה על הפרויקט נתנה לכם?**אני מרגיש שהבנתי יותר את כל הקטע של הצפנה.   
והכי חשוב שלמדתי עוד 3 שפות תכנות שלא ידעתי לפני HTML, CSS, JS.  
מצאתי בעצמי פתרונות לעוד כל מיני דברים כמו הנעילה של הצאטים ואיך ליעל את השרת של השיחות.  
למדתי קצת על אודיו בpython.

# 7. ביבליוגרפיה

1. <https://stackoverflow.com/questions/36894315/how-to-select-a-specific-input-device-with-pyaudio>  
עבור בחירת מיקרופון בשיחות, לא בטוח שאספיק לממש לזה GUI אז לא בטוח שזה יהיה בשימוש אבל זה כבר נמצא בקוד בתור comment.

2. <https://docs.python.org/3/library/tkinter.messagebox.html#module-tkinter.messagebox>  
עבור הצגת הודעות שגיאה בחלונית קטנה.

3. <https://stackoverflow.com/questions/51396841/how-to-change-a-python-thread-name-from-inside-the-thread-on-windows>  
עבור שינוי השם של הthread של הלקוחות שמתחברים לserver לאחר ביצוע login.

4. <https://github.com/python-eel/Eel/issues/395>  
הייתה לי בעיה כשאר השתמשתי עם הeel וניסיתי לפתוח file dialog ומצאתי את הבעיה הזאת שמישהו העלה והפתרון שם פתר גם לי את הבעיה.

היו עוד כל מיני דברים שנעזרתי בהם אבל לא שמרתי את הקישורים להכל.