# Group Chatter

## דניאל פנזיאס

| | |
|---|---|
| ישיבת בני-עקיבא אורות יהודה | אורות יהודה |
| תז: 328244959 | שם המנחה: שמעון סוויסה |
| שם החלופה: | 14.06.2023 |

# Group Chatter

## דניאל פנזיאס

# תוכן עניינים

# Group Chatter

## דניאל פנזיאס

# Group Chatter

## דניאל פנזיאס

## מבוא

## ייזום

### תיאור כללי

הפרויקט שלי הוא צ'אט קבוצתי/פתוח. התוכנה שיצרתי במסגרת הפרויקט, "Group Chatter", היא תוכנה מבוססת פייתון עם ממשק משתמש גרפי(GUI) המאפשרת תקשורת בין כמה משתמשים באמצעות שימוש בפרוטוקול UDP, socket עם הצפנת RSA א-סימטרית מתקדמת.

### מטרת הפרויקט

בחרתי לעבוד על פרויקט זה כיוון שרציתי ליצור צ'אט בסגנון מסוים שמאפשר לכמה לקוחות לדבר ביניהם בו זמנית. התוכנה שיצרתי במסגרת הפרויקט מאפשרת זאת ונותנת אפשרות לשיחה עם כמה אנשים בו זמנית.
המוטיבציה שמניעה אותי לעשות את הפרויקט הזה, היא הרצון להרחיב את הידע שלי בחיבור של שרת-לקוח וחיזוק היכולות הקיימות שלי בנושא.

### ליבה טכנולוגית

ליבת הפרויקט היא החיבור בין השרת ללקוחות והדיבור בניהם.

### טכנולוגיות עיקריות ושיקולים עיקריים

על מנת לממש את הליבה הטכנולוגית של הפרויקט שלי יהיה צורך ביצירת שרת - כל הפונקציות הנדרשות להחזיק שרת + פונקציות של חיבור עם לקוחות. לקוח - מחלקות ופונקציות שמנהלות את הלקוח ואת התקשורת שלו מול המשתמש, כמו כן גם GUI, ממשק יפה ונוח למשתמש. מסמך קוד המכיל את הפונקציונליות של הצפנת RSA שאותה אני מממש בפרויקט.
הסיבה בה בחרתי לממש את הפרויקט בשפת פייתון דווקא ולא שפה אחרת היא מכיוון שהשפה נוחה לשימוש במיוחד בכל מה שקשור לחיבור שרת-לקוח. ולא מצאתי שפת תכנות שבה יותר קל לעשות את זה.

# Group Chatter

## דניאל פנזיאס

### אתגרים טכנולוגיים ומקורות

האתגרים הטכנולוגיים הצפויים לי במימוש התוכנה הם בעיקר אתגרים מסוג פתירת שגיאות הקשורות לחיבור עצמו בין שרת ולקוח ופחות בשאר הקוד. מלבד מהשגיאות ייתכן שלא אבין מה אני צריך לעשות בחלק הזה של הקוד על מנת שהוא יעבוד וזה מקום שבו כנראה אשקיע המון זמן מהפרויקט עצמו.

מקורות מידע:
אתר Google וחבריו(StackOverflow, GeeksForGeeks, etc)
נוספים להם:
תיעוד פייתון

### סוגי משתמשים / קהל היעד

כל אדם יכול להשתמש במערכת, כל עוד יש לו את התוכנה שאותה ניתן להפעיל באמצעות הקבצים של הפרויקט. מתאים לכל אחד שרוצה להתנסות בהתקשרות של שרת לקוח והבנה של הנושא.

### דרישות חומרה

אין דרישות חומרה מלבד שיהיה למחשב יכולת להריץ שרת לקוח.

### פתרונות קיימים

קיימים אינספור אפליקציות ואתרי צ'אטים בין אנשים, אך מטרתו של הפרויקט לא להמציא דרך חדשה אלא לממש דבר קיים בדרך שונה הניתנת לבנייה על ידי אדם אחד.

# Group Chatter

## דניאל פנזיאס

## אפיון

### פיצ'רים ותהליכים עיקריים

פיצ'רים

**הרצת שרת - start_server:**
פיצ'ר זה מריץ את השרת על מנת שיוכלו להתחבר אליו וכדי שיהיה באפשרותו לחבר לקוחות שישלחו הודעות ביניהם. הפונקציות הרלוונטיות לפיצ'ר זה הם:
start_server.

**חיבור לשרת - connect:**
פיצ'ר זה אחראי על חיבור השרת עם הלקוח/ות שרוצים להשתמש בתוכנה.
החיבור נעשה על ידי UDP socket ומתבצע על המחשב המריץ את התוכנה(ניתן לשנות גם למחשבים אחרים בתנאי שמשתמשים בip וport נכונים. הפונקציות הרלוונטיות לפיצ'ר זה הם:
connect - גם בצד השרת וגם בצד לקוח.

**שליחת הודעה שרת ללקוח / לקוח לשרת - send_message:**
פיצ'ר זה אחראי על שליחת הודעות בין שרת ללקוח ובין לקוח לשרת כולל העברת ההודעות מהשרת לשאר הלקוחות. הפונקציות הרלוונטיות לפיצ'ר זה הם:
send_message - גם בצד השרת וגם בצד לקוח.

**קבלת הודעה מלקוח / שרת - receive_message:**
פיצ'ר זה אחראי על קבלת הודעות בין שרת ללקוח ובין לקוח לשרת. הפונקציות הרלוונטיות לפיצ'ר זה הם:
receive_message - גם בצד השרת וגם בצד לקוח.

**הצפנה ופענוח של הודעות בעזרת RSA - encrypt/decrypt:**
פיצ'ר זה אחראי להצפנה של כל ההודעות המועברות בין שרת ללקוח ובין לקוח לשרת לאחר התחברות בסיסית ומעבר של מפתחות ציבוריים על מנת שיוכלו להשתמש בצופן זה. הפונקציות הרלוונטיות לפיצ'ר זה הם:
encrypt_message - הצפנה של הודעה, במתכונת של טקסט.
decrypt_message - פענוח של ההצפנה על ההודעה שנשלחה בין אם שרת ללקוח ובין אם לקוח לשרת.

# Group Chatter

## דניאל פנזיאס

**אימות משתמש - authentication:**
פיצ'ר זה אחראי על אימות לקוח כאשר הוא מנסה להתחבר לצ'אט המשותף בין הלקוחות(לאחר חיבור לשרת), פועל באמצעות בדיקה של קיימות של שם לקוח במסד הנתונים, אם קיים אימות אם הסיסמא ואם לא הוספה של לקוח חדש במסד הנתונים. הפונקציות הרלוונטיות לפיצ'ר זה הם:
authentication - גם בצד שרת וגם לקוח, צד לקוח אחראי על התהליך, צד שרת מאמת נתונים ונותן אישור אם שם הלקוח והסיסמא שלו מתאימים למידע שיש במסד הנתונים.

**שליחת תוכן קובץ טקסט מלקוח - send_text_file:**
פיצ'ר זה מאפשר גישה לסייר הקבצים במחשב שממנו ניתן לבחור קובץ טקסט והתוכנה תקרא את תוכנו ותשלח לשרת שישלח לשאר הלקוחות. הפונקציות הרלוונטיות לפיצ'ר זה הם:
browse_file - הפונקציה מאפשרת גישה לסייר הקבצים ובחירת קובץ.
send_text_file - הפונקציה קוראת את הקובץ שנבחר ושולחת את תוכנו מוצפן לשרת.

## טכנולוגיות

| פיצ'ר/תהליך | טכנולוגיות ושפות תכנות | משאבים נדרשים ושירותים חיצוניים |
|---|---|---|
| **הרצת שרת - start_server** | python, UDP socket<br>נעשה שימוש בשני הטכנולוגיות, בפיתון על מנת להריץ את השרת בקוד ובUDP על מנת להעביר הודעות בין שרת ללקוח ולהפך. | ספריית socket וספריית threading בפייתון. |
| **חיבור לשרת - connect** | python, UDP socket<br>נעשה שימוש בשני הטכנולוגיות, בפיתון על מנת להתחבר אל השרת ובUDP על מנת להעביר הודעות בין שרת ללקוח ולהפך. | ספריית socket וספריית threading בפייתון. |
| **שליחת הודעה שרת ללקוח / לקוח לשרת - send_message** | python, UDP socket<br>נעשה שימוש בשני הטכנולוגיות, בפיתון על מנת לשלוח הודעות בין השרת ללקוח ולהפך באמצעות פרוטוקול UDP. | ספריית socket וספריית threading בפייתון וקובץ RSACode בו מוצפנות ההודעות. |

# Group Chatter

## דניאל פנזיאס

| | | |
|---|---|---|
| **קבלת הודעה מלקוח / שרת - receive_message** | python, UDP socket<br>נעשה שימוש בשני הטכנולוגיות, בפיתון על מנת לקבל הודעות בין השרת ללקוח ולהפך באמצעות פרוטוקול UDP. | ספריית socket וספריית threading בפייתון וקובץ RSACode בו מוצפנות ההודעות. |
| **הצפנה ופענוח של הודעות בעזרת - RSA encrypt/decrypt** | python, RSACode<br>נעשה שימוש בשני הטכנולוגיות, בפיתון על מנת לכתוב את הקוד שאחראי על ההצפנה ובמסמך RSACode נשמר הקוד. | קובץ RSACode בו מוצפנות ההודעות. |
| **אימות משתמש - authentication** | python, UDP socket<br>נעשה שימוש בשני הטכנולוגיות על מנת לקבל ולשלוח את הודעות האימות בין השרת ללקוח ולהפך באמצעות פרוטוקול UDP. | ספריית socket וספריית threading בפייתון וקובץ RSACode בו מוצפנות ההודעות. |
| **שליחת תוכן קובץ טקסט מלקוח - send_text_file** | python, UDP socket<br>נעשה שימוש בשני הטכנולוגיות, בפיתון על מנת לשלוח הודעות בין השרת ללקוח ולהפך באמצעות פרוטוקול UDP. | ספריית socket וספריית threading בפייתון וקובץ RSACode בו מוצפנות ההודעות וfiledialog המאפשר שימוש בסייר הקבצים. |

# Group Chatter

## דניאל פנזיאס

### לוח זמנים

איטרציה 1:

- עבודה על הקמת שרת שיעבוד מול כמה לקוחות רק ברמת ההתחברות.
- התחלת פיתוח קובץ צופן הRSA.
- פיתוח לקוח בסיסי ללא GUI.

איטרציה 2:

- המשך עבודה על השרת והתאמתו בהתאם לכל שלב.
- סיום פיתוח קובץ הRSA.
- התחלת פיתוח GUI.

איטרציה 3:

- המשך עבודה על הGUI.
- המשך עבודה על השרת.
- שדרוג הלקוח לרמה גבוהה יותר(יותר פעולות, עבודה בטוחה יותר וכו').
- בדיקות בסיסיות לעבודת שרת לקוח.

איטרציה 4:

- סיום שרת.
- סיום לקוח.
- סיום GUI.
- בדיקות חשובות ומקיפות על הפרויקט.

### ניהול סיכונים

סיכון 1: כאשר מריצים את השרת ואת הלקוח וסוגרים את הלקוח צריך לדאוג שהשרת לא יקרוס, בסיכון זה טיפלתי באמצעות שימוש בtry except ים שבעזרתם אם ישנה שגיאה אז התוכנית לא תקרוס.

סיכון 2: כאשר מריצים את השרת ואת הלקוח וסוגרים את השרת צריך לדאוג שהלקוח נסגר בצורה כמו שצריך ולא סתם יקרוס, בסיכון זה טיפלתי באמצעות שימוש בtry except ים שבעזרתם אם ישנה שגיאה אז התוכנית לא תקרוס.

סיכון 3: כאשר ישנה שגיאה במעבר המידע בין השרת ללקוח יש לדאוג שלא תהיה קריסה של אחד מהם, בסיכון זה טיפלתי באמצעות שימוש בtry except ים שבעזרתם אם ישנה שגיאה אז התוכנית לא תקרוס.

# Group Chatter

## דניאל פנזיאס

# מבנה / ארכיטקטורה

## מבט על

בחרתי לחלק את התוכנה לשני צדדים שונים, צד לקוח אשר לו יהיה את קובץ הRSA וקובץ הלקוח כמובן, וצד שרת שלו יהיה את קובץ הRSA גם ובנוסף את קובץ השרת כמובן.

**קובץ הRSA** - קובץ זה, אשר קיים גם אצל השרת וגם אצל הלקוח, מכיל את כל הפעולות אשר המערכת משתמשת בהם על מנת להצפין ולפענח את ההודעות שהמשתמשים והשרת רוצים להעביר ביניהם, הכוללים בתוכם: יצירת מפתחות הצפנה רנדומליים, הצפנת הודעה, פענוח הודעה. כמו כן קובץ זה מכיל מחלקה אחת ששמישה מבחוץ על מנת לשמור על אבטחה מרובה בשימוש הצופן.

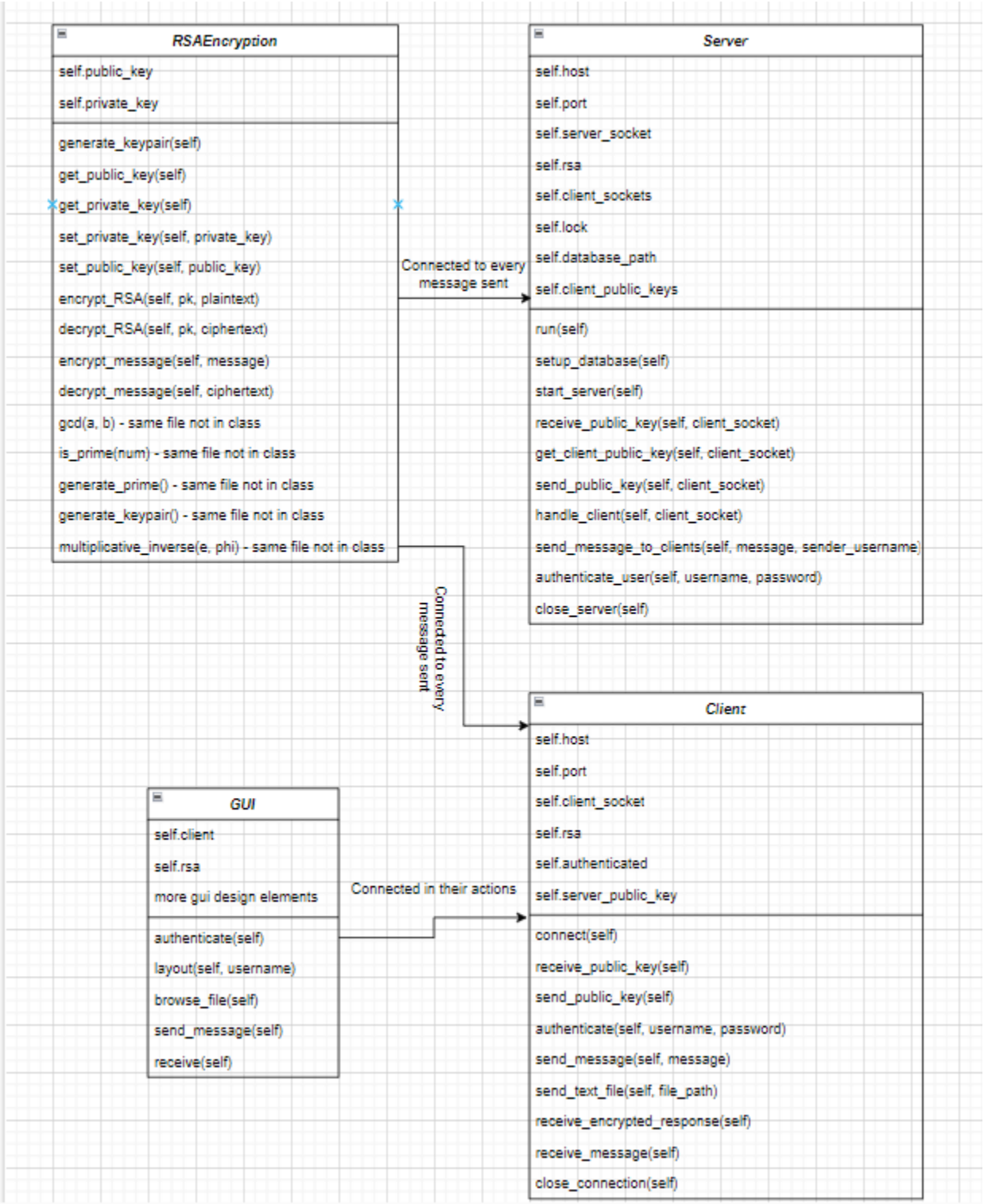
**קובץ הלקוח Client** - קובץ זה הוא הקובץ העיקרי של המערכת ומטרתו להתחבר לשרת שמחבר את הלקוח לכל שאר הלקוחות בצ'אט הכללי. בקובץ זה ישנם שתי מחלקות: מחלקת Client - המחלקה אחראית על כל הפונקציונליות של הלקוח כגון: התחברות לשרת, שליחת הודעה, קבלת הודעה, שליחת תוכן קובץ טקסט, אימות משתמש אל מול השרת וכו'. ומחלקת GUI - אחראית על כל הממשק הגרפי של המשתמש - שתי המסכים השונים, כניסת משתמש ומקום ההתכתבות, כפתורים, מראה נעים ונוח למשתמש.

**קובץ השרת Server** - קובץ זה מתנהל אל מול כל הלקוחות, בתחילה כל לקוח מתחבר אליו באמצעות מערכת אימות שמשתמשת בdatabase ולאחר מכן נשלחות כל ההודעות מהלקוחות לשרת והוא מעביר אותן לכל שאר הלקוחות המחוברים באותו עת. הפעולות העיקריות שיש בקובץ זה הם: חיבור לשרת, יצירת השרת, שליחת הודעות, קבלת הודעות, והתנהלות אל מול הלקוחות השונים.

# Group Chatter

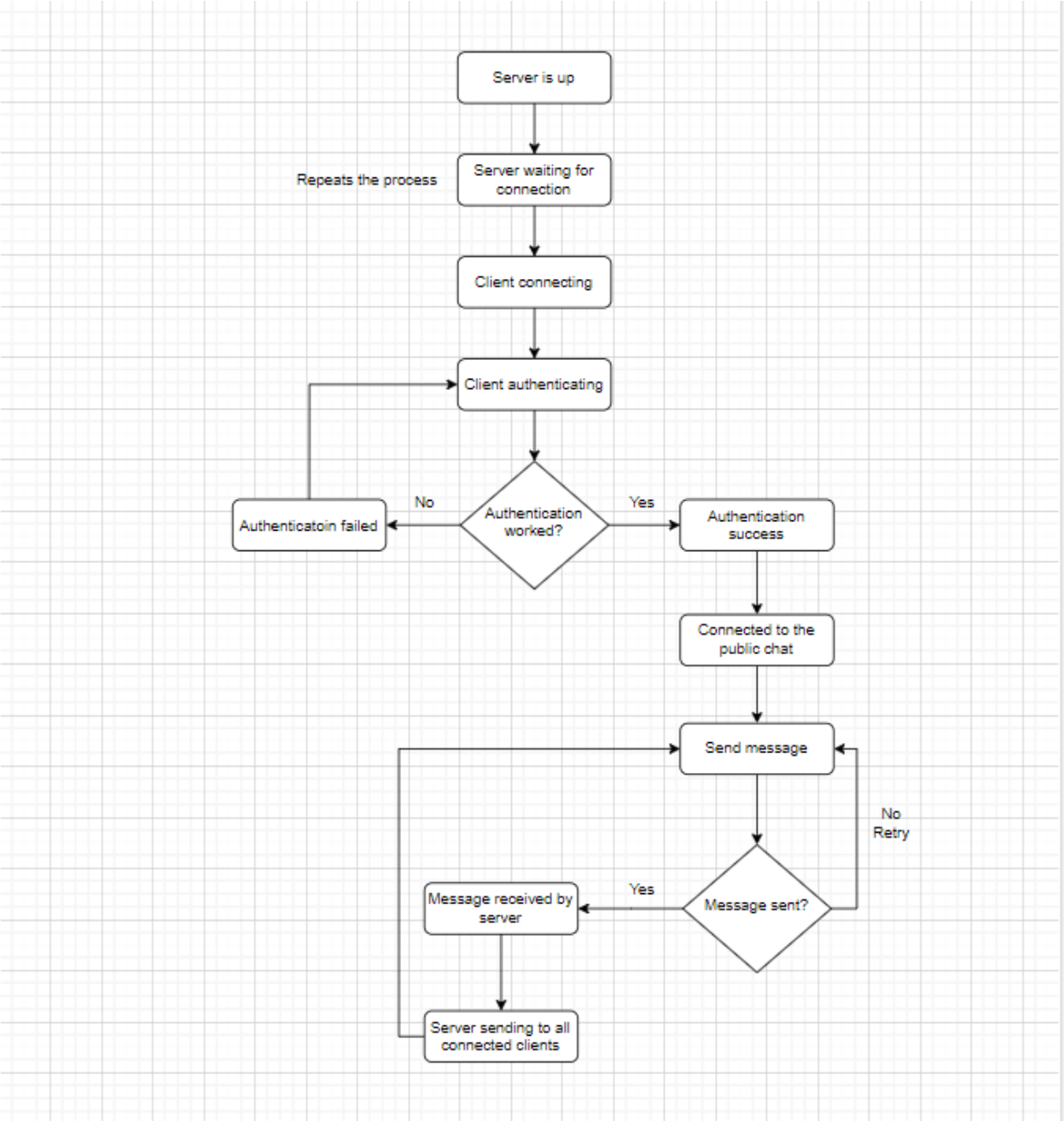
## דניאל פנזיאס

## תרשימים



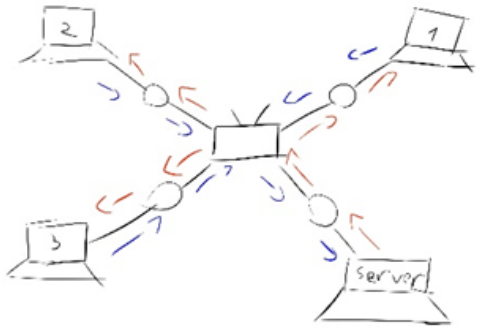
# Group Chatter

## דניאל פנזיאס

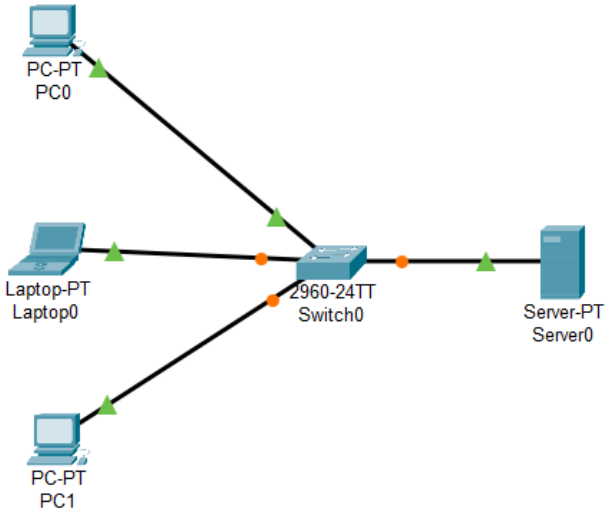


# Group Chatter

## דניאל פנזיאס







# Group Chatter

## דניאל פנזיאס

## פירוט רכיבי המערכת

### קובץ RSACode:

**get_public_key()**
קלט: אין.
פלט: מחזירה את הpublic_key של המשתמש.
הסבר: הפונקציה מחזירה את המפתח הציבורי של האובייקט הנוכחי(של המשתמש).

**get_private_key()**
קלט: אין.
פלט: מחזירה את הprivate_key של המשתמש.
הסבר: הפונקציה מחזירה את המפתח הפרטי של האובייקט הנוכחי(של המשתמש).

**set_private_key(private_key)**
קלט: private_key לשינוי בו משתמשים לערך הprivate_key.
פלט: אין.
הסבר: הפונקציה מקבלת מפתח פרטי ומשנה את המפתח הפרטי ששמור לאחד שהתקבל.

**set_public_key(public_key)**
קלט: public_key לשינוי בו משתמשים לערך הpublic_key.
פלט: אין.
הסבר: הפונקציה מקבלת מפתח ציבורי ומשנה את המפתח הציבורי ששמור לאחד שהתקבל.

**encrypt_RSA(pk, plaintext)**
קלט: מפתח ציבורי והודעה(string) לא מוצפנת.
פלט: הודעה מוצפנת - חוזרת כרשימה של מספרים.
הסבר: הפונקציה מקבלת מפתח ציבורי והודעה לא מוצפנת ואז מצפינה את ההודעה בעזרת המפתח הזה ומחזירה את ההודעה כרשימה של מספרים שלמים.

**decrypt_RSA(pk, ciphertext)**
קלט: מפתח פרטי והודעה מוצפנת.
פלט: הודעה(string) לא מוצפנת, מפוענחת.
הסבר: הפונקציה מקבלת מפתח פרטי והודעה מוצפנת ואז מפענחת את ההודעה בעזרת המפתח הזה ומחזירה את ההודעה כString.

# Group Chatter

## דניאל פנזיאס

**encrypt_message(message)**
קלט: הודעה(string) לא מוצפנת.
פלט: הודעה מוצפנת - חוזרת כרשימה של מספרים.
הסבר: הפונקציה מקבלת הודעה(string) לא מוצפנת, משתמשת בפונקציה encrypt_RSA על מנת להצפין את ההודעה ומחזירה רשימה של מספרים.

**decrypt_message(ciphertext)**
קלט: הודעה מוצפנת.
פלט: הודעה(string) לא מוצפנת, מפוענחת.
הסבר: הפונקציה מקבלת הודעה מוצפנת, משתמשת בפונקציה decrypt_RSA על מנת לפענח את ההודעה ומחזירה הודעה(Sting) מפוענחת.

**gcd(a, b)**
קלט: הפונקציה מקבלת שני מספרים שלמים- a, b.
פלט: : הפונקציה מחזירה מספר שלם המייצג את המכנה המשותף הכי גדול של a וb.
הסבר: הפונקציה מקבלת שני מספרים שלמים ומחזירה את המחלק המשותף הגדול ביותר שלהם.

**is_prime(num)**
קלט: הפונקציה מקבלת מספר שלם.
פלט: הפונקציה מחזירה- true אם המשתנה שנכנס אליה ראשוני ו- false אם אינו ראשוני.
הסבר: הפונקציה מקבלת מספר שלם ובודקת האם הוא מספר ראשוני או לא, אם כן מחזירה True, אם לא מחזירה False.

**generate_prime()**
קלט: אין.
פלט: הפונקציה מחזירה מספר ראשוני.
הסבר: הפונקציה עוברת בלולאה אין סופית ובה מייצרת מספר שלם רנדומלי באמצעות math.random ובודקת האם הוא מספר ראשוני באמצעות הפונקציה is_prime, כאשר נמצא מספר ראשוני הפונקציה עוצרת ומחזירה את המספר הראשוני.

# Group Chatter

## דניאל פנזיאס

**generate_keypair()**

קלט: אין.

פלט: מפתח ציבורי ופרטי בצורת tuple.

הסבר: הפונקציה מייצרת מפתח ציבורי ופרטי באמצעות יצירת שני מספרים ראשוניים באמצעות הפונקציה generate_prime ולאחר מכן משתמשת בפעולות generate_prime, gcd, multiplicative_inverse על מנת שיתווספו עוד שני מספרים להם שישמשו כמפתח ציבורי ופרטי שמשלימים אחד את השני.

**multiplicative_inverse(e, phi)**

קלט: שני משתנים שלמים.

פלט: שארית החלוקה של המספר הראשון בשני.

הסבר: הפונקציה מקבלת שני מספרים שלמים ובעזרתם מוצאת את 'המכנה המשותף', החלק המשותף בין המפתח הפרטי לציבורי ומחזירה את שארית החלוקה של המספרים האלה אחד בשני.

# Group Chatter

## דניאל פנזיאס

### קובץ Server:

**run()**
קלט: אין.
פלט: אין.
הסבר: הפונקציה מריצה את צד השרת באמצעות יצירת מפתח ציבורי ופרטי - הפונקציה generate_keypair ואז מפעילה את בסיס הנתונים, אם הוא קיים כבר פותחת אותו ואם לא מייצרת חדש - הפונקציה setup_database, ולאחר מכן מריצה את השרת - הפונקציה start_server.

**setup_database()**
קלט: אין.
פלט: אין.
הסבר: הפונקציה מקימה את בסיס הנתונים במידה והוא לא קיים כבר, אם הוא קיים היא פותחת אותו.

**start_server()**
קלט: אין.
פלט: אין.
הסבר: הפונקציה מתחילה את השרת ורצה בלולאה אין סופית על קבלת לקוחות ומפרידה אותם לתהליכונים שונים שמריצים את הפונקציה handle_client שמטפלת בכל לקוח ולקוח כך שהלקוחות ירוצו במקביל.

**receive_public_key(client_socket)**
קלט: הסוקט של הלקוח שאיתו מדבר השרת.
פלט: אין.
הסבר: הפונקציה אחראית על קבלת המפתח הציבורי מהמשתמש על מנת שיוכלו לתקשר בניהם.

**get_client_public_key(client_socket)**
קלט: הסוקט של הלקוח שאיתו מדבר השרת.
פלט: מפתח ציבורי של הלקוח הנוכחי.
הסבר: הפונקציה מחזירה את המפתח הציבורי מתוך מילון המפתחות הציבורים על ידי שימוש במפתח של כל לקוח לקוח שהוא הסוקט שלהם.

**send_public_key(client_socket)**
קלט: הסוקט של הלקוח שאיתו מדבר השרת.
פלט: אין.
הסבר: הפונקציה שולחת את המפתח הציבורי של השרת אל הלקוח על מנת שהם יוכלו לתקשר.

# Group Chatter

## דניאל פנזיאס

**handle_client(client_socket)**
קלט: הסוקט של הלקוח שאיתו מדבר השרת.
פלט: אין.
הסבר: הפונקציה אחראית על כל מה שקשור בין תקשורת לשרת ולקוח ומאפשרת ממנה לדבר עם כל שאר הלקוחות בצ'אט הציבורי. הפונקציה עובדת בכמה שלבים, היא מתחילה בהחלפת מפתחות ציבוריים בין השרת ללקוח ואז היא ממשיכה לשלב האימות שם משתמש וסיסמא של הלקוח ולאחר מכן אם האימות צולח אז היא שולחת ללקוח הודעה שהאימות צלח ולאחר מכן מאפשרת שיחה עם שאר המשתמשים המחוברים לצ'אט הציבורי. אם האימות כשל אז היא שולחת שהאימות כשל ומחכה לאימות חדש.

**send_message_to_clients(message, sender_username)**
קלט: הודעה לשליחה לשאר הלקוחות המחוברים ושם השולח.
פלט: אין.
הסבר: הפונקציה מקבלת הודעה(String) ואת שם השולח של ההודעה ואז מעבירה לכל הלקוחות המחוברים לצ'אט הציבורי את ההודעה בצורה מוצפנת, ההודעה מוצפנת לכל לקוח על פי המפתח הציבורי שלו ואז נשלחת אליו שם הוא מפענח את ההודעה ורואה אותה.

**authenticate_user(username, password)**
קלט: שם משתמש וסיסמא של מי שמנסה להתחבר.
פלט: אימות צלח או כשל(String).
הסבר: הפונקציה מקבלת שם משתמש וסיסמא ומנסה לאמת אותו בעזרת המידע שיש בבסיס הנתונים. במידה והמשתמש קיים אז היא בודקת אם הסיסמא תואמת למידע בבסיס הנתונים, אם לא היא מחזירה הודעת אימות לא צלח ואז המשתמש ישלח אימות חדש. אם היא כן תואמת למידע בבסיס הנתונים אז היא מחזירה הודעת אימות צלח והמשתמש ממשיך לצ'אט הציבורי.
במידה והמשתמש לא היה קיים בבסיס הנתונים אז הוא מתווסף לשם כמשתמש חדש וממשיך לצ'אט הציבורי.

**close_server()**
קלט: אין.
פלט: אין.
הסבר: הפונקציה סוגרת את השרת כאשר הוא מסיים את תפקידו ונסגר.

# Group Chatter

## דניאל פנזיאס

### קובץ Client:

Client class:

**connect()**
קלט: אין.
פלט: Bool שמסמן אם החיבור צלח או כשל.
הסבר: הפונקציה אחראית על חיבור לשרת, אם החיבור צולח אז היא מחזירה הודעת True, ואם החיבור כושל אז היא מחזירה הודעת False.

**receive_public_key()**
קלט: אין.
פלט: Bool שמסמן אם החיבור צלח או כשל.
הסבר: הפונקציה אחראית על קבלת המפתח הציבורי של השרת על מנת שהלקוח יהיה יכול לתקשר עם השרת.

**send_public_key()**
קלט: אין.
פלט: אין.
הסבר: הפונקציה אחראית על שליחת המפתח הציבורי של הלקוח אל השרת על מנת שיוכלו לתקשר ביניהם.

**authenticate(username, password)**
קלט: שם משתמש וסיסמא.
פלט: אימות צלח או כשל(String).
הסבר: הפונקציה מקבלת שם משתמש וסיסמא ומנסה לאמת אותם אל מול השרת, במידה והאימות מצליח היא שולחת חזרה הודעת אימות צלח ואז ממשיכים לצ'אט הציבורי, במידה והאימות נכשל אז היא מחזירה הודעת אימות כשל וחוזרת על התהליך.

**send_message(message)**
קלט: הודעה לשליחה לשרת.
פלט: אין.
הסבר: הפונקציה אחראית על שליחת הודעות לשרת, היא מקבלת הודעה(String) - המשתנה message, מצפינה אותו בעזרת המפתח הציבורי של השרת ואז שולחת לשרת את ההודעה המוצפנת.

# Group Chatter

## דניאל פנזיאס

**send_text_file(file_path)**

קלט: נתיב קובץ הטקסט אותו אנחנו קוראים ושולחים את תוכנו לשרת.

פלט: אין.

הסבר: הפונקציה אחראית על שליחת תוכן של קובץ טקסט לשרת, היא עושה זאת על ידי קבלת נתיב קובץ הטקסט, קריאת המידע בתוכו ושליחתו לשרת באמצעות הפונקציה send_message.

**receive_encrypted_response()**

קלט: אין.

פלט: מחזיר את ההודעה המוצפנת לאחר שעשה לה decode.

הסבר: הפונקציה מקבלת את ההודעה המוצפנת מהשרת, עושה לה decode ואז מחזירה אותה על מנת שימשיכו לפענח אותה בעזרת צופן הRSA.

**receive_message()**

קלט: אין.

פלט: מחזיר את ההודעה שהתקבלה מהשרת.

הסבר: הפונקציה אחראית על קבלת הודעות מהשרת באמצעות הפונקציה receive_encrypted_response, לאחר מכן מפענחת את ההודעה באמצעות צופן הRSA ולבסוף מחזירה את ההודעה המפוענחת ללקוח על מנת שיוכל 'להדפיס' אותה אצלו בצ'אט הציבורי.

**close_connection()**

קלט: אין.

פלט: אין.

הסבר: הפונקציה סוגרת את החיבור לשרת בכל מקרה, בין אם הלקוח סוגר את עצמו ובין אם השרת נסגר ואז היא סוגרת את החיבור ביניהם.

# Group Chatter

## דניאל פנזיאס

GUI class:

**authenticate()**

קלט: אין.

פלט: אין.

הסבר: הפונקציה אחראית על כל התהליך מהחיבור לשרת עד סיום האימות אל מולו, בתחילה חיבור לשרת, לאחר מכן אימותו של הלקוח אל מול השרת ואז פתיחה של התהליכון האחראי על המסך של הצ'אט הציבורי.

**layout(username)**

קלט: שם משתמש שאותו כותבים בראש המסך של הצ'אט הציבורי של כל לקוח.

פלט: אין.

הסבר: הפונקציה אחראית על סידור המסך של הצ'אט הציבורי כולל כל הכפתורים שיש בו והפונקציות הקשורות אליו.

**browse_file()**

קלט: אין.

פלט: אין.

הסבר: הפונקציה אחראית על האפשרות של בחירת קובץ מסייר הקבצים(קובץ טקסט) על מנת שנקרא אותו ונשלח לשרת.

**send_message()**

קלט: אין.

פלט: אין.

הסבר: הפונקציה קוראת את מה שכתוב בשורת ההודעה בצ'אט הציבורי וקוראת לפונקציה send_message שנמצאת במחלקה Client על מנת לשלוח את ההודעה הזאת לשרת.

**receive()**

קלט: אין.

פלט: אין.

הסבר: הפונקציה מקבלת את ההודעה שנשלחה מהשרת באמצעות receive_message במחלקה Client ואז 'מדפיסה' את ההודעה הזאת בצ'אט הציבורי של הלקוח.

# Group Chatter

## דניאל פנזיאס

### database:

זהו המבנה הבסיסי של כל משתמש שנשמר בdatabase.

| | | |
|---|---|---|
| Id - INTEGER primary key | Username - Text | Password - Text |

כל הפעולות אל מול הdatabase מתבצעות בשרת שמאמת לקוחות שמנסים להתחבר, יוצר את בסיס הנתונים וסוגר אותו כאשר צריך.

### סקירת מודלים:

socket - מודול שמספק פונקציות המשמשות לתקשורת בין שרת ללקוח.
tkinter - מודול שמספק את כל ההיבט העיצובי של התוכנית - GUI.
threading - מודול שמספק את האפשרות לבצע כמה תהליכים במקביל.
RSA_encryption - קובץ מיובא שכולל בתוכו את כל הפונקציונליות של צופן הRSA בו משתמשים בפרויקט.

### סקירת מחלקות:

**RSAEncryption** - אחראית על ההצפנה של RSA בפרויקט.
פירוט תכונות:
public_key - מפתח ציבורי המשמש להצפנה של הודעות בצופן RSA.
private_key - מפתח פרטי המשמש לפענוח של הודעות בצופן RSA.
פעולות במחלקה:
מפורטות במבנה / ארכיטקטורה, קובץ RSACode.

**Server** - אחראית על התקשורת בין הלקוחות באמצעות multithreading.
פירוט תכונות:
host - הIP עליו רץ השרת.
port - הפורט עליו רץ השרת.
server_socket - הסוקט עליו רץ השרת.
rsa - אובייקט מסוג RSAEncryption שמנהל את הצפנת RSA של השרת אל מול הלקוחות.
client_sockets - מילון של סוקטים של לקוחות, מפתח הוא שם משתמש, ערך הוא סוקט של אותו לקוח.
lock - משתנה שמטרתו לנהל את פעולת התהליכונים במקומות שיש סכנה שכמה תהליכונים יפעלו בו זמנית.
database_path - הנתיב שבו נשמר בסיס הנתונים ששומר את המידע על הלקוחות.
client_public_keys - מילון של מפתחות ציבוריים של משתמשים, מפתח הוא סוקט של לקוח, ערך הוא המפתח הציבורי שלו.

# Group Chatter

## דניאל פנזיאס

פעולות במחלקה:
מפורטות במבנה / ארכיטקטורה, קובץ Server.

**Client** - אחראית על ההצפנה של RSA בפרויקט.
פירוט תכונות:
host - הIP עליו רץ הלקוח.
port - הפורט עליו רץ הלקוח.
client_socket - הסוקט של הלקוח אל מול השרת.
rsa - אובייקט מסוג RSAEncryption שמנהל את הצפנת RSA של הלקוח אל מול השרת.
authenticated - משתנה בוליאני שמסמן האם הלקוח עבר אימות אל מול השרת או לא.
server_public_key - המפתח הציבורי של השרת שאיתו מצפינים הודעות לשרת מהלקוח.
פעולות במחלקה:
מפורטות במבנה / ארכיטקטורה, קובץ Client, Client class.

**GUI** - אחראית על ההצפנה של RSA בפרויקט.
פירוט תכונות:
client - אובייקט מסוג Client שמפעיל את הלקוח אל מול השרת.
rsa - משתנה שהוא client.rsa שהוא אובייקט מסוג RSAEncryption שמנהל את הצפנת RSA של הלקוח אל מול השרת.
פעולות במחלקה:
מפורטות במבנה / ארכיטקטורה, קובץ Client, GUI class.

# Group Chatter

## דניאל פנזיאס

# מדריך למשתמש

### קבצי מערכת:

RSACode - ראה פירוט במבנה / ארכיטקטורה - מבט על.
Client - ראה פירוט במבנה / ארכיטקטורה - מבט על.
Server - ראה פירוט במבנה / ארכיטקטורה - מבט על.

### סביבה נדרשת:

פייתון 3.10

### הפעלת מערכת:

הפעלת שרת:

פתיחת קובץ השרת והרצתו.
ניתן להפעיל את השרת על ידי הכנסת פקודה זו לcmd -
python (code_path) Server.py self 12345
code_path - הנתיב לאיפה שנמצא הקוד על המחשב.

הפעלת לקוח:

פתיחת הלקוח והרצתו.
ניתן להפעיל את השרת על ידי הכנסת פקודה זו לcmd -
python (code_path) Client.py
code_path - הנתיב לאיפה שנמצא הקוד על המחשב.

### שימוש במערכת:

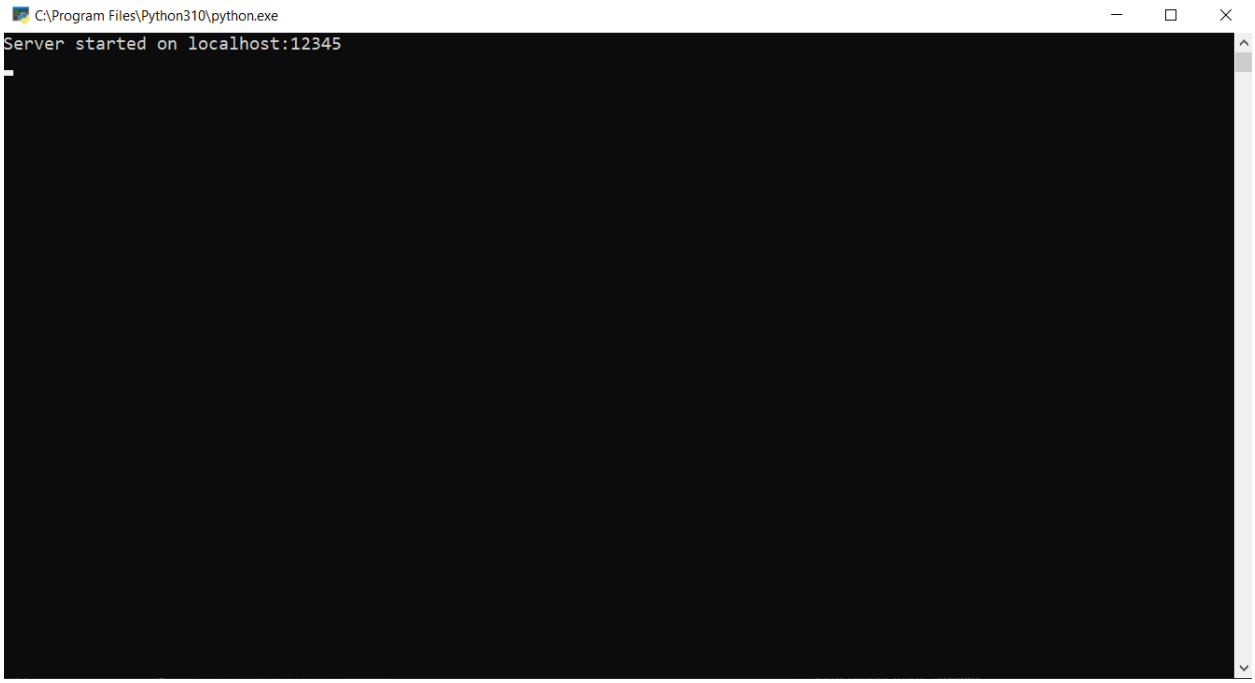
לאחר הפעלת השרת ופתיחת לקוח כלשהו יש להתחבר דרך מסך הlogin, אם יש לך משתמש כבר תכניס אותו ואת הסיסמא שלו ואם לא הכנס משתמש חדש עם סיסמתו ותחובר לצ'אט הציבורי.

# Group Chatter

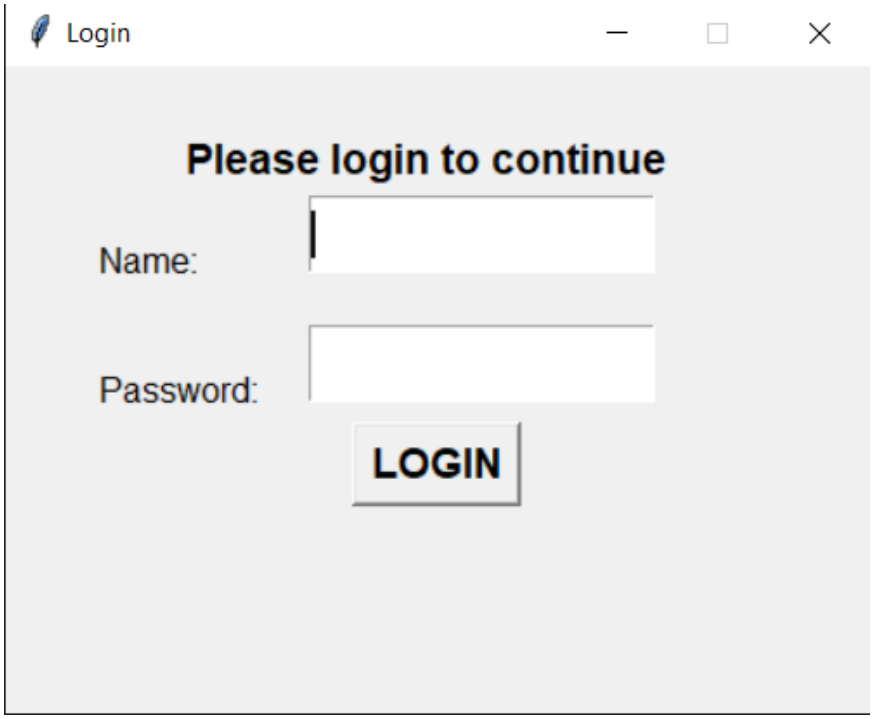
## דניאל פנזיאס

### תמונות להמחשה:

שרת מורץ:



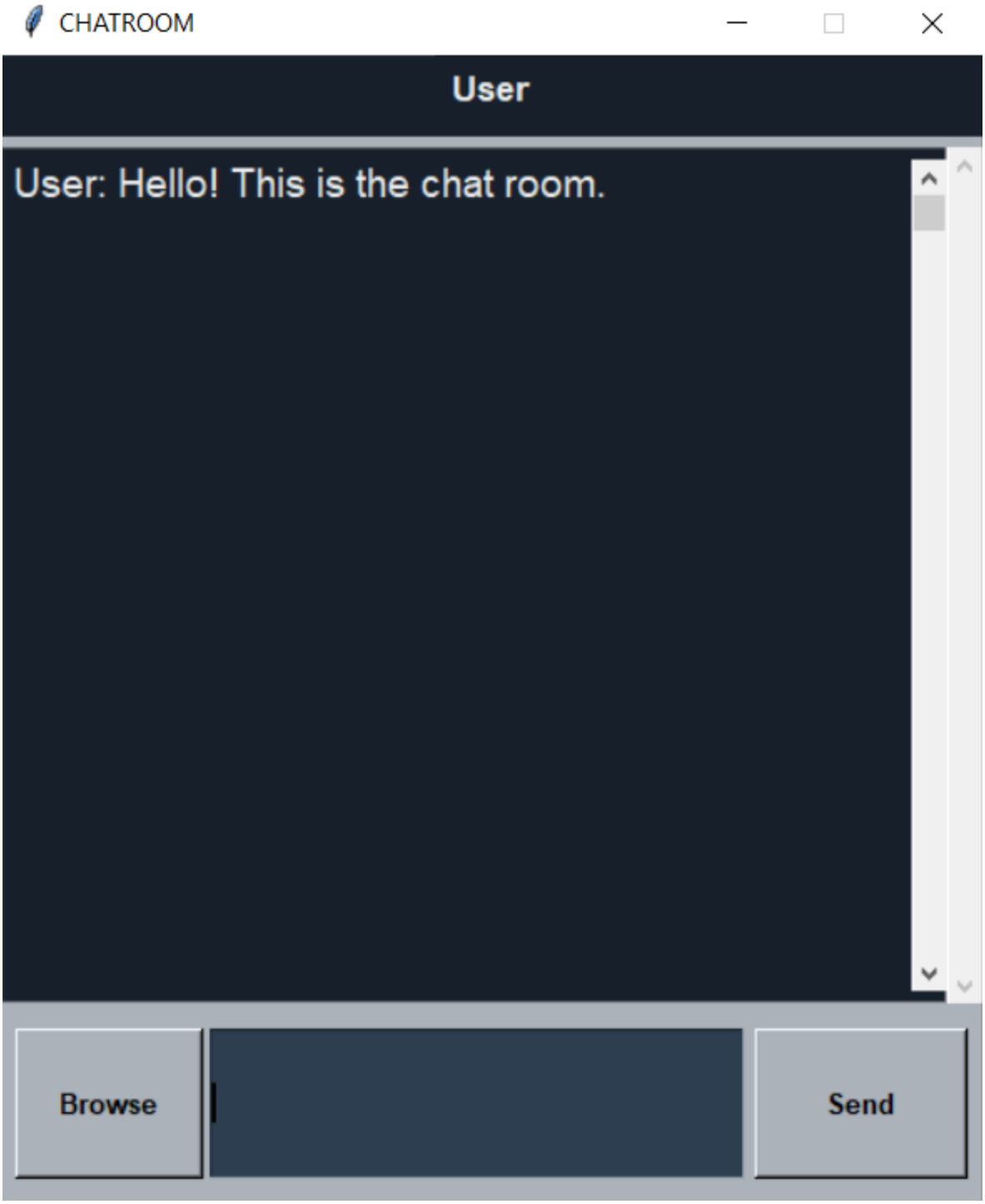

מסך כניסה לקוח:



# Group Chatter

## דניאל פנזיאס

מסך צ'אט ציבורי לקוח:



# Group Chatter

## דניאל פנזיאס

## סיכום אישי / רפלקציה

במהלך פרויקט זה חיזקתי כמה יכולות שלי בעולם התכנות ובניהן: תכנות בשפת פייתון, תכנון חכם יותר של המערכת כאשר מתכנתים, כתיבה מהירה יותר של קוד, הבנה של תוכניות מבוססות שרת-לקוח בצורה רחבה יותר ופתרון שגיאות בצורה יעילה יותר. במהלך העבודה על הפרויקט נתקלתי בכמה קשיים הקשורים למוטיבציה להמשיך ולתכנת ולנסות ולפתור את הבעיות בקוד גם כאשר יש אלף ואחד באגים אחד אחרי השני. הדרך בה מצאתי את המוטיבציה להמשיך ולתכנת גם כאשר הכל לא עובד ולא כיף היא להגיד לעצמי שזהו רק שלב אחד שאם אותו אעבור אז אגיע לשלב הבא ומשם נתקדם, מה שנקרא "נעבור את הגשר כשנגיע אליו". התובנות אותן אני לוקח מהעבודה על פרויקט זה הן שאף פעם לא לוותר כאשר מתמודדים מול קוד שנראה אולי טיפה קשה מכיוון שתמיד יש דרך לפתור בעיה ותמיד אחרי זמן מסויים מבינים מה קורה בכל מקום. תובנה נוספת שאני לוקח מהעבודה על הפרויקט היא שכאשר  אני רוצה לעבוד על משהו ונהנה ממנו אז אני יכול לעבוד עליו שעות רבות ברצף ולהצליח בו ברמה מאוד גבוהה. בראייה לאחור אחרי שסיימתי את הפרויקט יכול להיות שהייתי מחליט על פרויקט אחר כדי שאני אהנה יותר מהתהליך, וכמעט בוודאות שהייתי מתחיל לעבוד על הקוד בצורה שונה שלא תכניס אותי למלא שגיאות שצריך לפתור כל רגע בגלל איזושהי פונקציה אחת או אחרת שכתבתי. לבסוף אני רוצה להגיד תודה לכל מי שעזר לי במהלך הפרויקט, אם זה בעזרה בהבנה של איזשהו חלק בקוד שאני לא מצליח להבין מה הבעיה בו, ואם זה בתמיכה ודחיפה לעבודה יותר טובה על הפרויקט במהלך כל הדרך.

# Group Chatter

## דניאל פנזיאס

## ביבליוגרפיה


GeeksForGeeks

Python

Stackoverflow

# Group Chatter

## דניאל פנזיאס

## **נספחים**

כל הקבצים מצורפים בdrive שבו יש את הפרויקט - קוד וקבצי תמך לקוד, תמונות, תרשימים וכל מה ששומש.