

פרויקט רשות

תאריך הגשה: 23/01/2026

מגישים:

שלו מאור - 322600057
נעם אגאי - 324172055

שם קבוצה: SHN

קישור לקוד המקור: <https://github.com/agnoam/hit-basic-networking-project>

חלק א':

1. הכנת התשתיות: ייצרת תעבורת ה-HTTP:

בשלב הראשון, הוגדר תוכן ההודעות בשכבה היישום תוך תמקדות בפרוטוקול HTTP.

מקור הנתונים:

- קובץ CSV בשם "input_http_N" נוצר.
- קובץ ה-CSV מכיל 20 שורות המדמאות אינטראקטיבית מלאה בין ללקוח לשרת.
- תעבורה זו כוללת בקשות GET למשאים שונים, תגובה שרת מתאימים, והודעות לסגירת התחשרות.

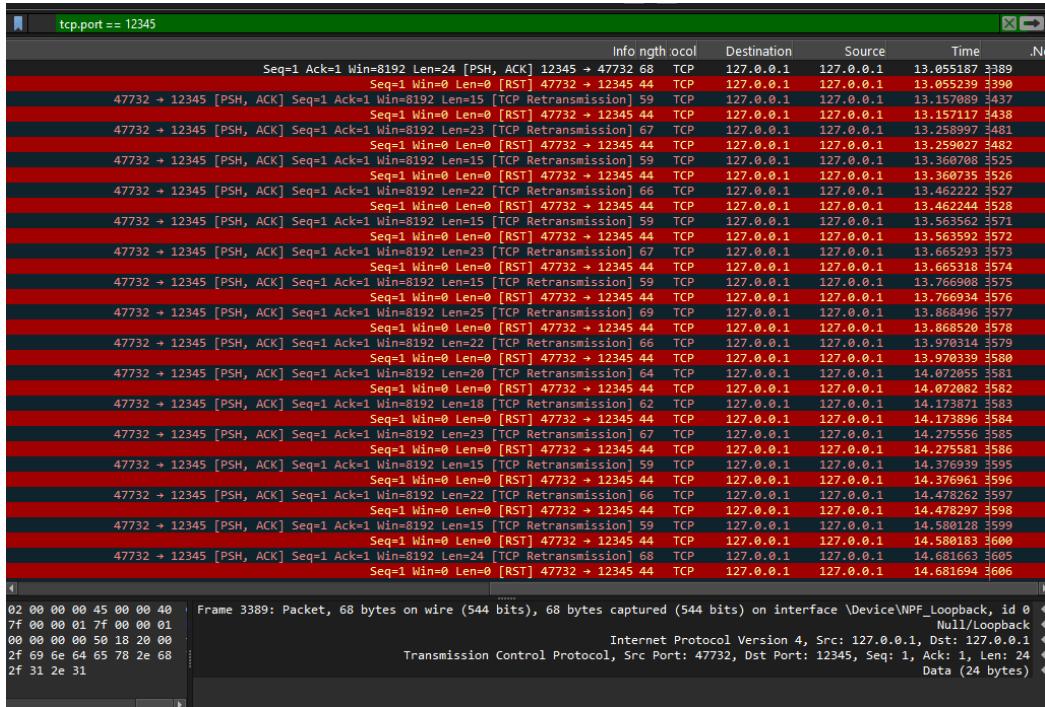
2. תהליך Capture Process :

תיעוד התעבורה וניתוחה בוצע באמצעות תוכנת Wireshark על פי הצעדים הבאים:

- שימוש במסנן ייעודי:** "tcp.port == 12345" כדי לבדוק את החבילות הרלוונטיות לפרויקט מתוך כלל תעבורת המחשב.
- תהליך הליכידה הופעל ב-Wireshark במקביל להרצת קוד ה-Jupyter Notebook. הקוד קרא את 20 ההודעות מה-CSV ושלח אותן ב프וטוקול TCP/IP המדמה את שכבות הרשת.

3. שימוש ב-Wireshark וניתוח הנטוונים:

- צילום מסך מ-Wireshark עם הסקן הרלוונטי:



- במהלך הרצת קובץ ה-CSV ב-Jupyter, בוצעה סימולציה של שילוח הודעות משלבת היישום לראשונה. מכיוון שהליקוי והשרות הופעל על אותו מחשב, כתובות המקור והיעד בעמודות ה-Source ו-hDestination היו זהות. כתובות (127.0.0.1) - המיצגת כתובות Loopback וירטואלית בתוך המחשב.
- צילום המסך לעיל מציג את ניסיונות התקשרות בין הליקוי לשרת.

- בוצע שימוש לבנייה מלאכותית - Gemini ליצירת ה-.CSV.

חלק ב':

שלב 1: הסבר כללי על המערכת:

- מערכת הziat בנויה מימי חלקיים כפי שלמדנו בקורס, צד שרת הצד לקוח. על הצד השרת להאזין ל-`requests` SMB צד לקוח. בזרה הבסיסית, רק הלקוח יוזם את החיבור לשרת. ולשרת אין דרך "להודיע" ללקוח על שינויים שקרו מבלי שהלקוח בקש.
- על מנת ליצור מערכת ziat אמינה ויעילה, יש צורך בפרוטוקול תקשורת שיתמוך בסוג תקשורת שבו השירות יכול לעדכן את הלקוחות על דעת עצמו. מה שלא מתאפשר בפרוטוקולים הבסיסיים. לכן בפרויקט נשתמש בפרוטוקול ה-`web socket`.

שלב 2: הסבר על מימוש הקוד:

בחרנו בשפת Python על מנת לפתח את הפROYIKT. שפה זו מעניקה יכולת הריצה על כל סוג מערכת הפעלה, אך מצריכה התקנה מראש. שפת Python היא שפה סקריפטית שלא מתאפשרת הקימפול שלה קורה בזמן ריצה. על מנת לשמר את הפROYIKT פשוט, מימשנו גם את הצד הלקוח כ-"תוכנה" בעלת GUI. השימושנו בעזרת `tkinter`.

שלב 3: צד השירות (`server.py`):

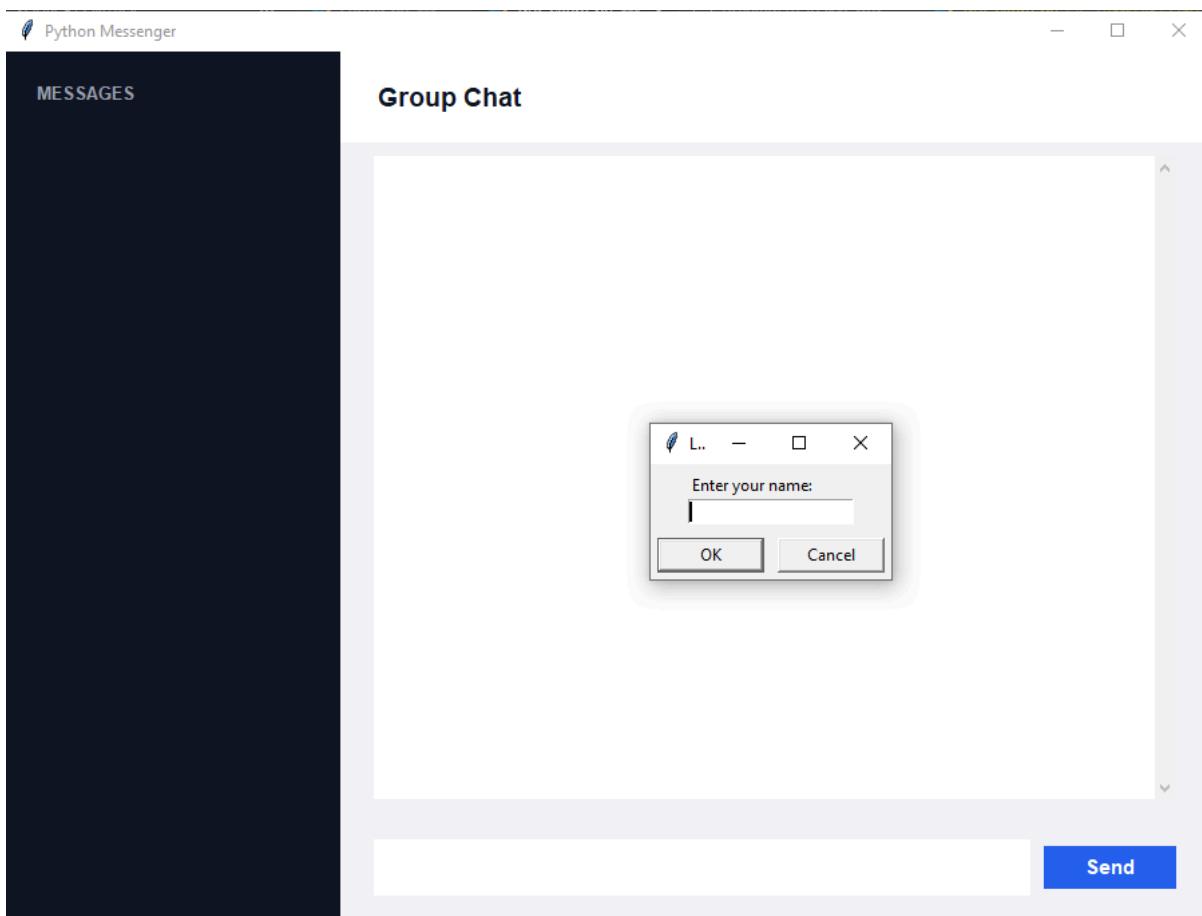
מנהל את החיבורם למשתמשים ושולח הודעות בין המשתמשים בזמן אמת, את פעולות הziat, ניהול המשתמשים והפרדה בין שיחות קבוצתיות לפרטיות.

להלן פירוט של כל סוג events הקיימים במערכת:

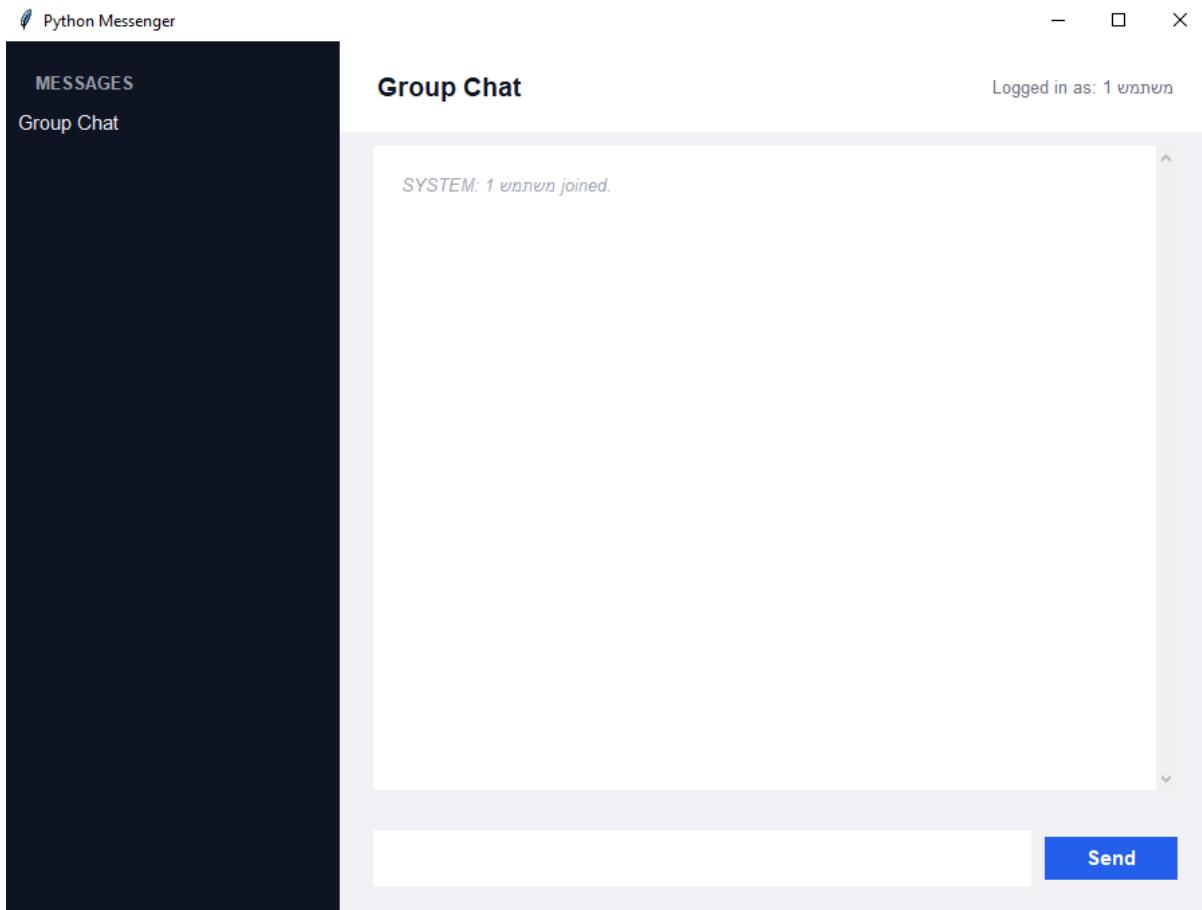
- חיבור משתמש חדש** - סוג הודעה שהשרת שולח ללקוח מיד עם התconnexionו ומכליה את מספר ה-ID שלו. (ההודעה זו נשלחת רק פעם אחת)
- רשימת משתמשים** - לאחר חיבור משתמש חדש ו/או ניתוק של משתמש קיימ, תשליח הודעה שמתחליה ב: "USER_LIST" לאחריה יופיעו שמות המשתמשים בהפרדת פסיק ביןיהם.
- הודעות הטרופות ועזיבה:** ההודעות אלו נשלחות לכל הקבוצה בפורמט {name} SYSTEM joined/left. הן משמשות לעדכן המשתמשים על פעילותם בחדר.
- כיבוי שירות:** כאשר השירות נסגר (על ידי פקודה EXIT או Ctrl+C), הוא מדיר לכולם את הודעה SERVER_SHUTTING_DOWN בצד SYSTEM: כדי שהלקוחות יוכל להתרנק בצורה מסודרת.
- הודעת קבוצה:** כל הודעה שהלקוח שולח אינה מתחילה בפקודה מיוחדת, השירות מפיץ אותה לכל שאר המשתמשים. השירות מוסיף לה הודעה זו את שם השולח בפורמט {name} message.
- הודעה פרטית (PRIVATE):** כאשר לקוח שולח הודעה המתחילה ב-/target {target}, השירות מזהה את שם היעד ושולח את ההודעה רק ל-`Socket` של אותו משתמש.

שלב 4: צד לקוח (`gui_client.py`):

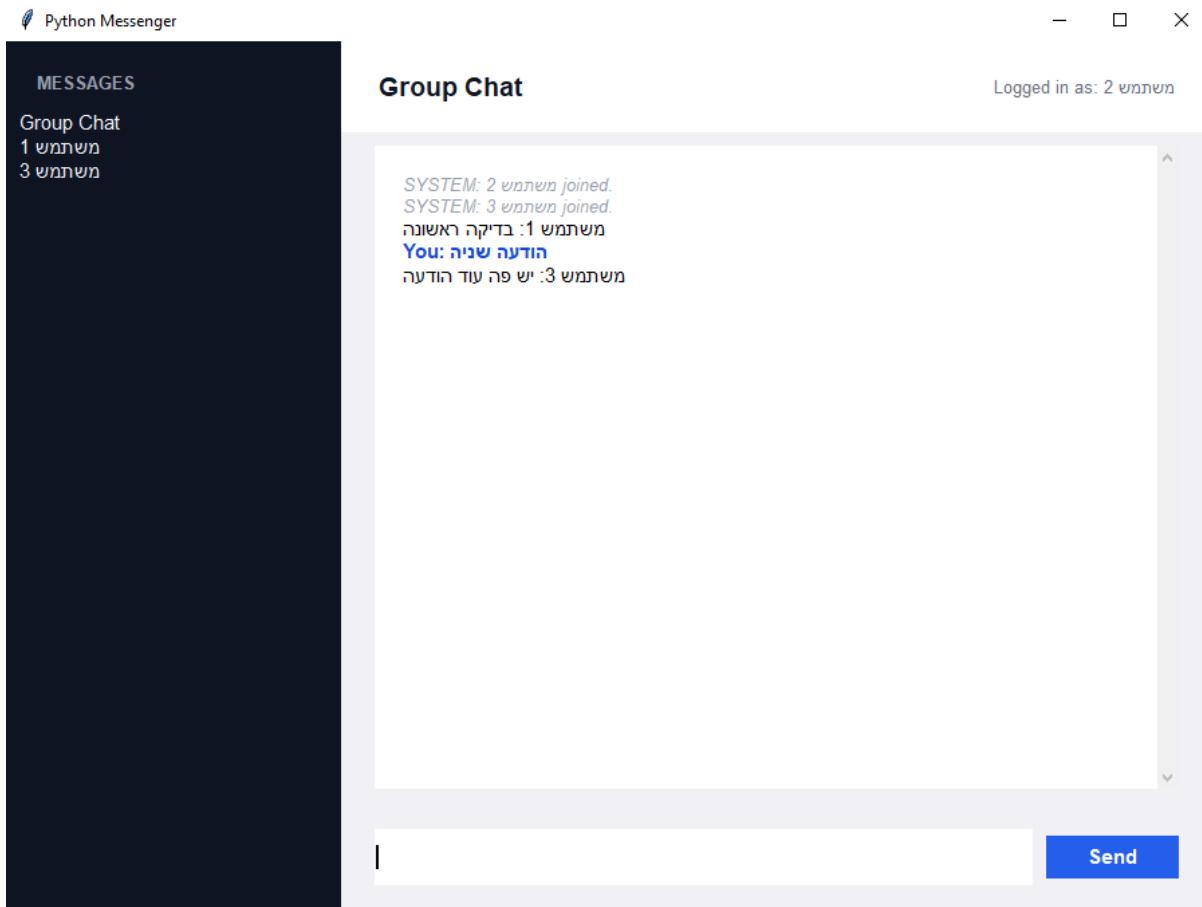
קובץ זה הוא "תוכנת" הziat. זה סקריפט בשפת Python שבו ממומשת תצוגה בעזרת `tkinter`. לשמורים את התוכנה, היא מבקשת מהמשתמש ל כתוב את השם שלו (על השם להיות שם ייחודי) (תמונה להמחשה).



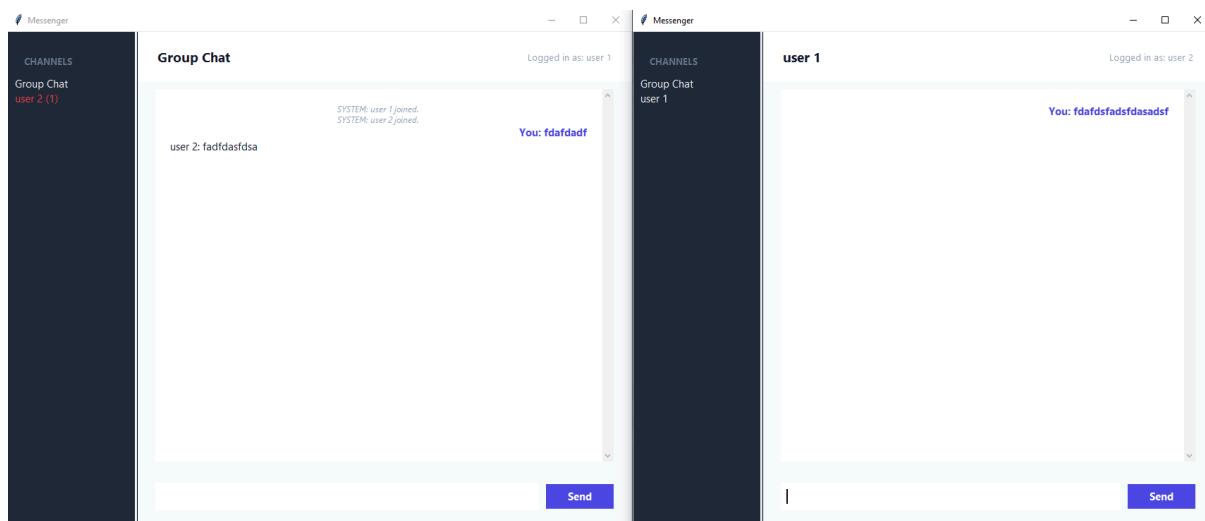
לאחר שהמשתמש בחר את השם המסן יראה כה:



לאחר שיש מספר משתמשים מחוברים, המשתמשים יכולים להתכתב ביניהם (בקבוצה):



בנוסף, כל משתמש יכול לשלוח הודעה למשתמש אחר עלי ידי בחירת השם מצד שמאל.



בתמונה זו רואים שליחת הודעה של פרטי משתמש אחד אל השני. כאשר המשתמש השני אינו צפה עדין בהודעה, השם של המשתמש יראה במצב אדום.

1. מקביליות וניהול תהליכיים (Concurrency)

כאן השאלה היא איך המערכת מבצעת מספר פעולות בו-זמןית.

- בצד השרת: איך נבנה את לולאט ה-accept כך שהוא לא תחסום את יכולת של השרת לשדר הודעות למשתמשים קיימים?
- בצד הלקוח: מדוע הקוד המKeySpec להודעות (receive_messages) חייב לרוץ ב-thread? נפרד מהלולאה הראשית (mainloop)?
- סירה נקייה: איך משתמש בסיגנלים (כמו SIGINT או פקודת EXIT) כדי לסגור את כל ה-Sockets הפתוחים בצורה מסודרת מוביל להשאיר "תהליכי רפאים" במחשב?

הודעות פרטיות: איזה "סימן" (כמו הפקודה /msg) נשתיל בתוך המחרוזת כדי שהשרת ידע לנוטב את ההודעה למשתמש יחיד ולא לכל הקבוצה?

עדכוניים חיים: איך נבנה את הודעה LIST_USER כך שהמשתמש הראשי ידע לרענן את רשימת אנשי הקשר באופן אוטומטי.

2. ניתוח רכיבי הרשות (Network Traffic)

כל התעבורה בלוג זה היא מסוג TCP. זהו פרוטוקול "מכoon חיבור" המבטיח שכל הודעה תגיע בשילמותה ובסדר הנכון.

- לחיצת היד המשולשת (Three-way Handshake): מופיעה בתחילת כל חיבור חדש (שורות 131-133 ו-235-237).
 - [SYN]: הלקוח מבקש להתחבר.
 - [SYN, ACK]: השרת מאשר ומבקש לסתור.
 - [ACK]: הלקוח מאשר סופית. מרגע זה הציגור פתוח.
- [PSH, ACK]: סימון (Flag) המעיד על העברת מידע בפועל (Data Transfer). ה-PSH אומר למערכת להעביר את המידע מיד לאפליקציה ולא לחכota במאפר.
- Length: גודל המידע בבייטים. זהו המدد הטוב ביותר לדעת מה נשלח (שם קצר, הודעה ארוכה וכו').

3. ניתוח שלבי האפליקציה לפי שורות

שלב א': התחברות "User 1" (שורות 131-141)

- שורות 131-133: הקמת החיבור הפיזי של המשתמש הראשון (פורט מקור 51891).
- שורה 134 (Len=2): השרת שולח ללקוח את ה-ID שלו. הגודל (2 בייטים) מותאים למחרוזת כמו "0\1".
- שורה 136 (Len=8): הלקוח שולח לשרת את פרטי ה-Handshake שלו. הגודל מותאים למחרוזת כמו ".User 1:0".
- שורה 138 (Len=17): השרת שולח את רשימת המשתמשים המעודכנת (חו 1\User_LIST).
- шורה 140 (Len=23): השרת מופיע הודעה מערכת: \!joined SYSTEM: User 1 joined.

שלב ב': התחברות "User 2" (233-242)

- שורות 233-235: המשמשו שני מתחבר (פורט מקור 51900).
- שורה 236 (Len=2): השרת שולח ID חדש ("1\ch").
- שורה 238 (Len=8): המשמשו שני נרשם כ-User 2:1-2.
- שורה 240 (Destination port 51891): שם לב! השרת שולח הודעה ללקוח הראשוני CDI לעדכן אותו שימושה חדש הטרף.

4. הקשר בין הקוד לתעבורה

הלוג מוכיח שהמנגנים שהטמענו עובדים:

- **הפרדת הודעות (\ch)**: ניתן לראות שההודעות נשלחות בנפרד (שורות עוקבות עם Length קטן). ה-\ch בסוף כל הודעה הוא זה שמאפשר ללקוח לדעת ששורות 138 הסתיימה ושורות 140 היא הודעה חדשה.
- **Broadcasting (שידור)**: שורה 240 היא ההוכחה שהשרת יודע לניהל מספר סוקטים בו-זמןית – הוא מקבל מידע מייד מפורט 51900 (User 2) ומעביר אותו לפорт 51891 (User 1).
- **יעילות**: רוב החבילות הן בגודל של 44-70 בייטים. זהו "תקורה" (Overhead) נמוכה מאוד, מה שהופך את היצוא למהיר מאוד בזמן אמת.

5. סיכום סטטיסטי של הלוג:

- **סך הכל חבילות**: 50 חבילות TCP.
- **כתובות**: הכל פנימי (127.0.0.1), מה שמעיד על בדיקה מקומית.
- **סטטוס אפליקציה**: הלוג מדווח הצלחות של שני משתמשים ותחילת סנסרין רשיימת משתמשים ביניהם. בנוסף, קיימות רשומות שמייצגות העברת הודעות בין המשתמשים.

6. שימוש ב-AI:

כשכתבנו את הפרויקט, השתמשנו ב-AI על מנת המשיך ולחזור לעומק נושאים תוך כדי פיתוח הפרויקט. אלו חלק מן ה-prompts המיצגים שבעזרתם למדנו ופיתחנו.

שלב 1: הבנת ארכיטקטורת הרשת (Sockets)

שלב זה מתמקד באיך המחשבים "מדוברים" זה עם זה.

- **פרופרומפט להבנת הפרטוקול:** "מהו תפקידו של SOCK_STREAM ו-INet_AF בקוד הרשת? איך הם מבטיחים שההודעות יגיעו לייעד ב次ורה אמינה?"
- **פרומפט לפתרון בעיות תקשורת:** "הסביר מדוע הוספנו את המנגנון של תו שורה חדשה (ach) בסוף כל הודעה בשרת וכי צד ה-buffer בклиינט מונע מהודעות 'להידבק' אחת לשניה".

שלב 2: ריבוי תהליכיים ומקביליות (Threading)

הבנייה איך הרשת מטפל בהרבה אנשים והקלינט מתקשר ומציג בו-זמן.

- **פרומפט לניהול משתמשים בשרת:** "כיצד הרשת משתמש ב-threading.Threads כדי לנהל שיחות עם מספר לקוחות בו-זמנית מבליל מוביל שהתוכנה תיעזר?"
- **פרומפט לסגירה מסודרת:** "איך הרשת מצליחה להפסיק גם לחרובים חדשים וגם לפקודת 'EXIT' בטרמינל בו-זמןית בעזרת daemon=True?"

שלב 3: ממשק משתמש וניהול מצב (GUI & State)

איך בונים את הצד הוויזואלי ומנהלים את המידע (מי שלח למי).

- **פרומפט להפרדת שיחות:** "כיצד אפשר להשתמש במשתנה מסווג מילון כדי להפריד בין הודעות פרטיות לבין ה-Chat, כך שכל שיחה תופיע רק בחלון הנכון?"
- **פרומפט למערכת התראות:** "באילו דרכים אפשר להשתמש, על מנת לדעת מתי להוסיף מספר ליד שם משתמש ומתי לאפס אותו בעת ניהול משתמשים בפרטוקול מבוסס web "socket

שלב 4: למידה מעשית (יצירה מחדש)

בקשות שיעזרו לך לכתוב קוד דומה你自己.

- **פרומפט לכנתיבת שרת בסיסי:** "הציג לי דוגמא לקוד פיתון בסיסי לשרת TCP שיודיע רק לקבל חיבור אחד ולהציג 'Hello World' ללקוח, כדי שאבין את הבסיס לפני הוסף GUI".
- **פרומפט ללמידה Tkinter:** "הסביר לי איך משתמשים ב-`config`-tag בתוך `ScrolledText` כדי לישר טקסט לימין או לשמאל, ואיך זה עוזר לייצור UI".

ניהול הודעות: כיצד נקבע שהודעה לא "תיחתר" או "תידבק" להודעה אחרת בגלל מגבלת MESSAGE_SIZE_IN_BYTES

זיהוי משתמשים: איזו לחיצת יד (Handshake) קריכה להתבצע מיד עם החיבור כדי שהשרת ידע לשירות Socket לשם משתמש ספציפי?