

פרויקט רשתות

תאריך הגשה: 23/01/2026

מגישים:

שלו מאור - 322600057

נועם אגאי - 324172055

שם קבוצה: SHN

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

חלק א':

1. הכנת התשתית: יצירת תעבורת ה-HTTP:

בשלב הראשון, הוגדר תוכן ההודעות בשכבת היישום תוך התמקדות בפרוטוקול HTTP.

מקור הנתונים:

- קובץ CSV בשם "SHN_http_input" נוצר.
- קובץ ה-CSV מכיל 20 שורות המדמות אינטראקציה מלאה בין לקוח לשרת.
- תעבורה זו כוללת בקשות GET למשאבים שונים, תגובות שרת מתאימות, והודעות לסגירת ההתקשרות.

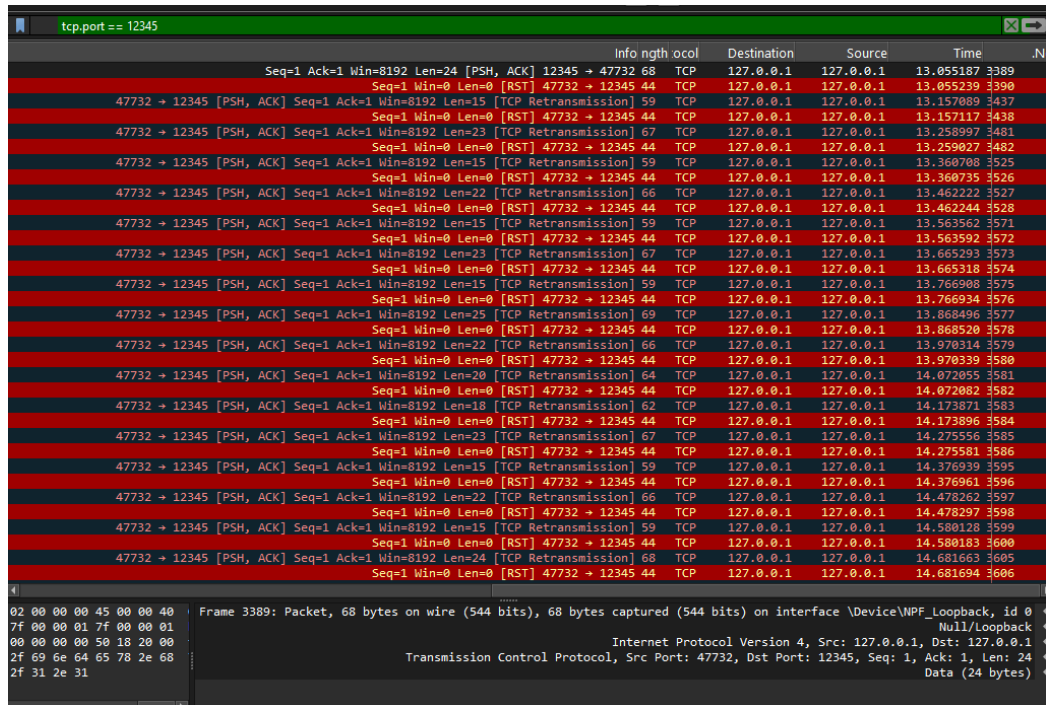
2. תהליך Capture Process:

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

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

3. שימוש ב-Wireshark וניתוח הנתונים:

- צילום מסך מ-Wireshark עם הסנן הרלוונטי:



- במהלך הרצת קובץ ה-CSV ב-Jupyter, בוצעה סימולציה של שליחת הודעות משכבת היישום לרשת. מכיוון שהלקוח והשרת הופעלו על אותו מחשב, כתובות המקור והיעד בעמודות ה-Source וה-Destination היו זהות. כתובת (127.0.0.1) - המייצגת כתובת Loopback וירטואלית בתוך המחשב.
- צילום המסך לעיל מציג את ניסיונות התקשורת בין הלקוח לשרת.
- בוצע שימוש בבינה מלאכותית - Gemini ליצירת ה-CSV.

חלק ב':

שלב 1:

הסבר כללי על המערכת:

- מערכת הצ'אט בנויה משני חלקים כפי שלמדנו בקורס, צד שרת וצד לקוח. על הצד השרת להאזין ל-requests שמבוצעים על ידי הלקוחות. בצורה הבסיסית, רק הלקוח יוזם את החיבור לשרת. ולשרת אין דרך "להודיע" ללקוח על שינויים שקרו מבלי שהלקוח ביקש.
- על מנת ליצור מערכת צ'אט אמינה ויעילה, יש צורך בפרוטוקול תקשורת שיתמוך בסוג תקשורת שבו השרת 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 הפרוייקט. שפה זו מעניקה יכולת הרצה על כל סוג מערכת הפעלה, אך מצריכה התקנה מראש. שפת Python היא שפה סקריפטית שלא מתקפלת. הקימפול שלה קורה בזמן ריצה. על מנת לשמור את הפרוייקט פשוט, מימשנו גם את צד הלקוח כ-"תוכנה" בעלת GUI. השתמשנו בעזרת tkinter.

שלב 3:

צד השרת (server.py):

מנהל את החיבורים למשתמשים ושולח הודעות בין המשתמשים בזמן אמת, את פעולת הצ'אט, ניהול המשתמשים והפרדה בין שיחות קבוצתיות לפרטיות.

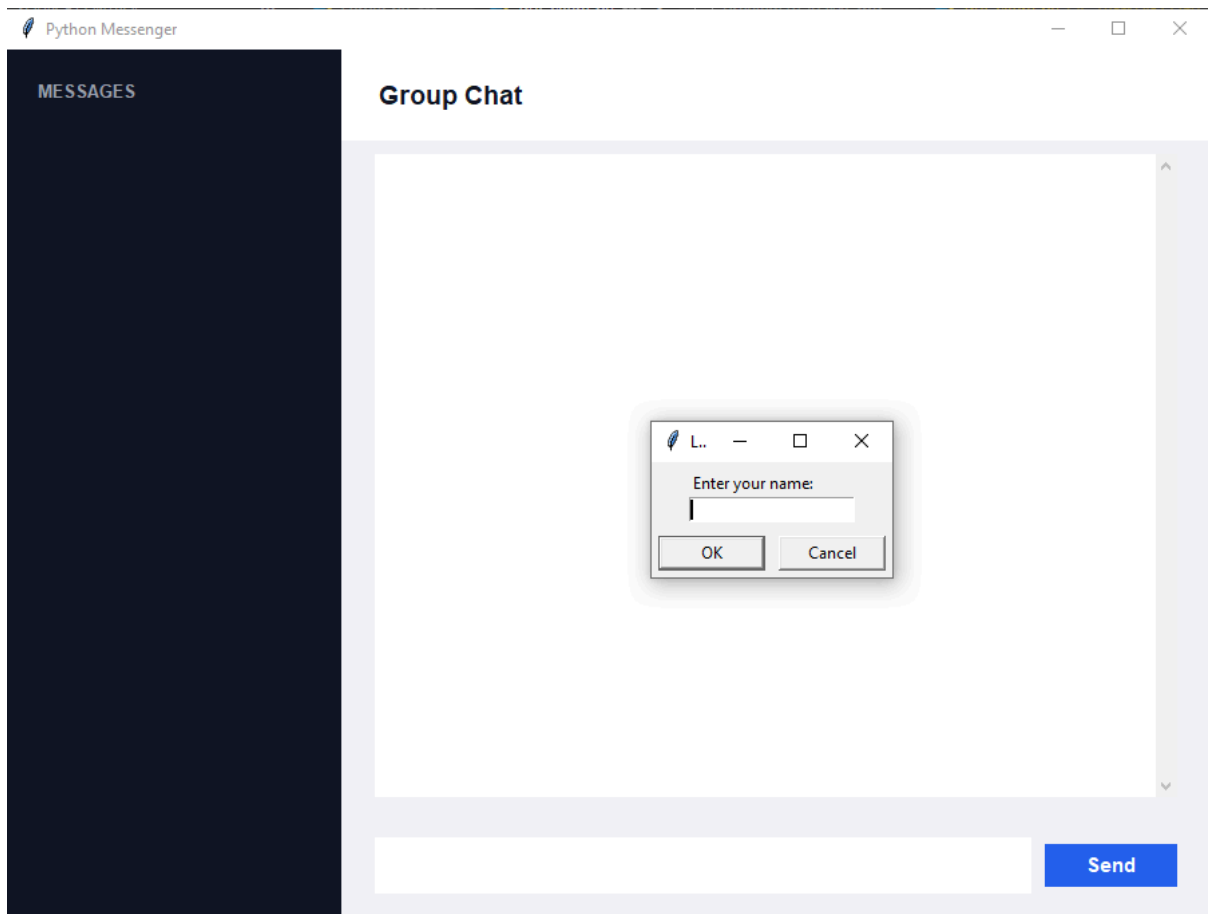
להלן פירוט של כל סוגי ה-events הקיימים במערכת:

- **חיבור משתמש חדש** - סוג הודעה שהשרת שולח ללקוח מיד עם התחברותו ומכילה את מספר ה-ID שלו. (הודעה זו נשלחת רק פעם אחת)
- **רשימת משתמשים** - לאחר חיבור משתמש חדש ו/או ניתוק של משתמש קיים, תשלח הודעה שמתחילה ב: "USER_LIST" לאחריה יופיעו שמות המשתמשים בהפרדת פסיק ביניהם.
- **הודעות הצטרפות ועזיבה:** הודעות אלו נשלחות לכל הקבוצה בפורמט {name} SYSTEM joined/left. הן משמשות לעדכון המשתמשים על פעילות בחדר.
- **כיווי שרת:** כאשר השרת נסגר (על ידי פקודת EXIT או Ctrl+C), הוא משדר לכולם את ההודעה SYSTEM: SERVER_SHUTTING_DOWN כדי שהלקוחות יוכלו להתנתק בצורה מסודרת.
- **הודעת קבוצה:** כל הודעה שהלקוח שולח ואינה מתחילה בפקודה מיוחדת, השרת מפיץ לכל שאר המשתמשים. השרת מוסיף להודעה זו את שם השולח בפורמט {name}: {{message}}.
- **הודעה פרטית (PRIVATE):** כאשר לקוח שולח הודעה המתחילה ב- msg {target}/, השרת מזהה את שם היעד ושולח את ההודעה רק ל-Socket של אותו משתמש.

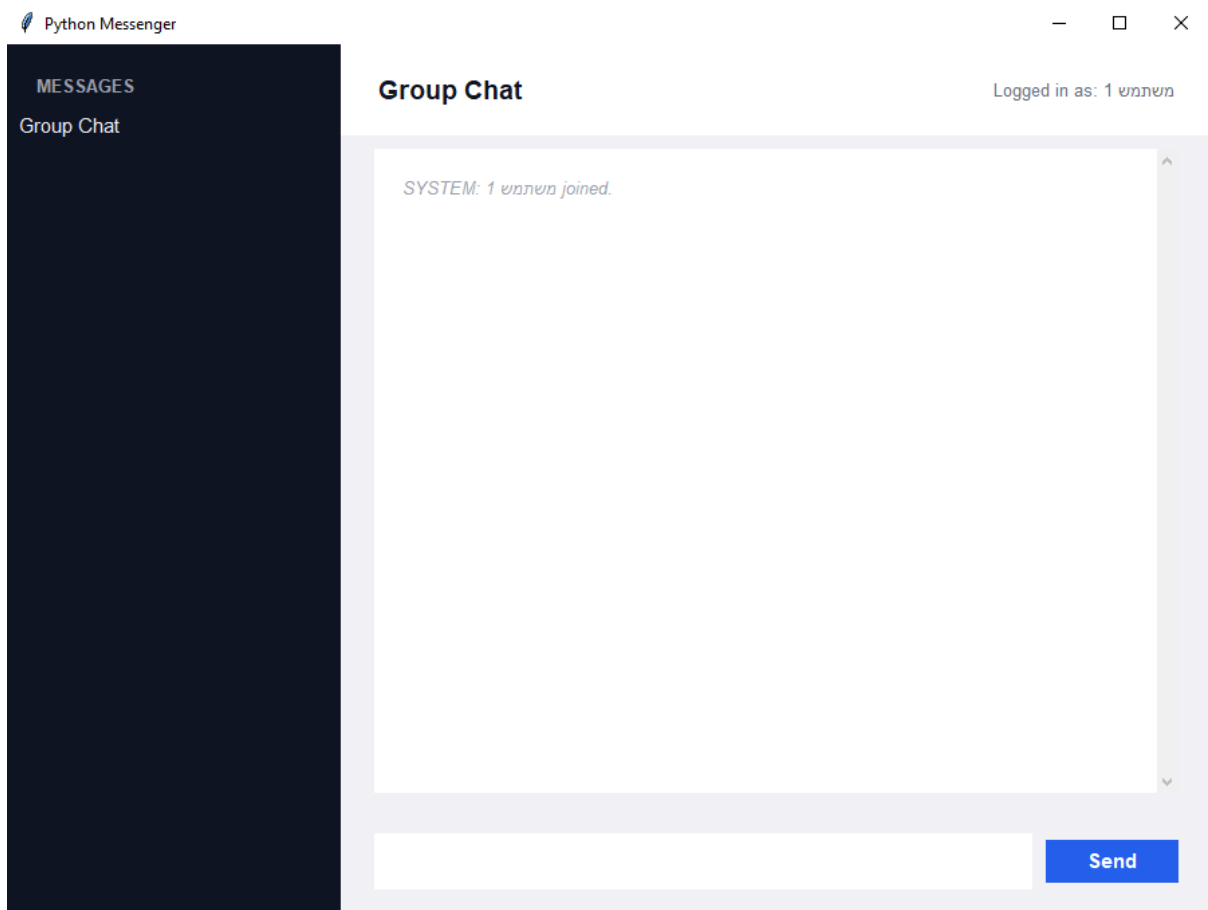
שלב 4:

צד לקוח (gui_client.py):

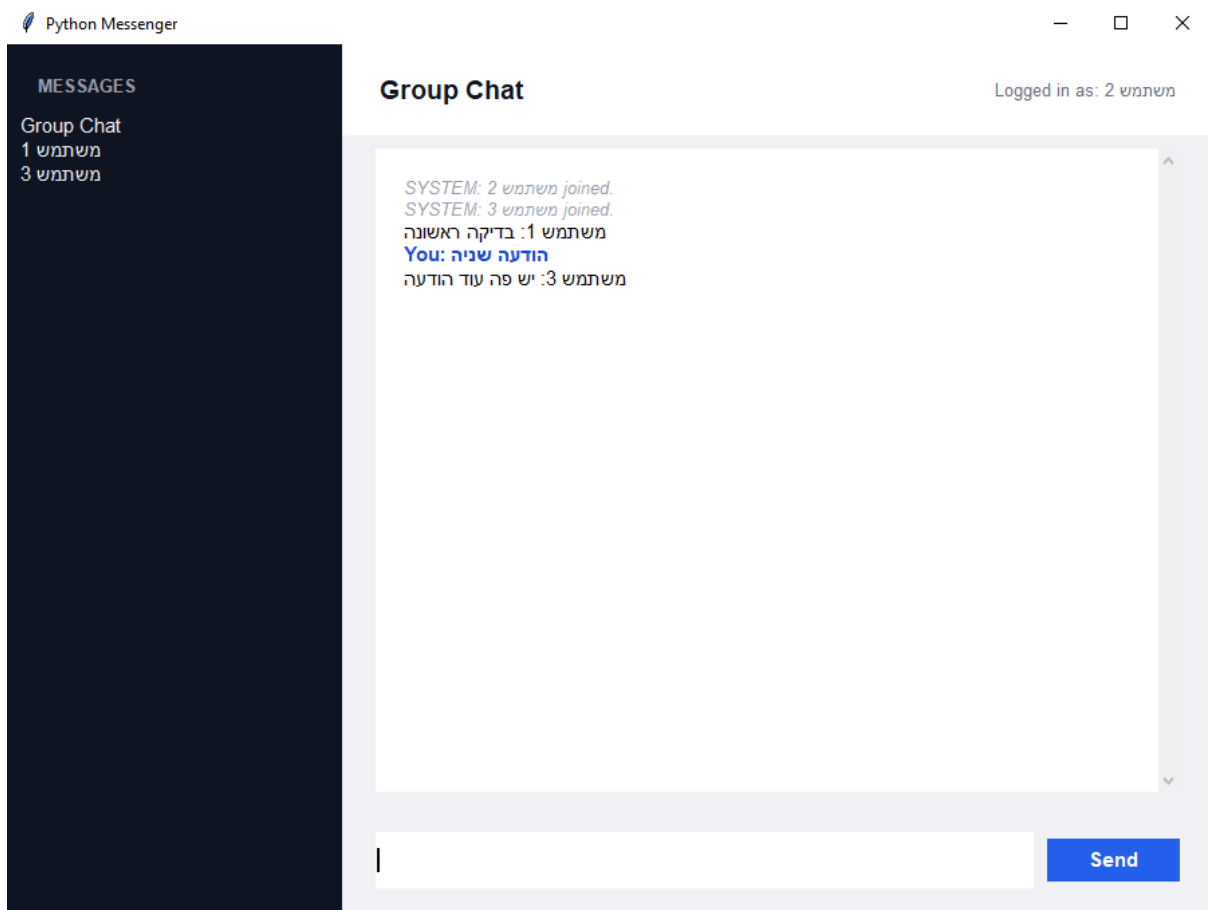
קובץ זה הוא "תוכנת" הצ'אט. זהו סקריפט בשפת Python שבו ממומשת תצוגה בעזרת tkinter. כשמריצים את התוכנה, היא מבקשת מהמשתמש לכתוב את השם שלו (על השם להיות שם ייחודי) (תמונה להמחשה).



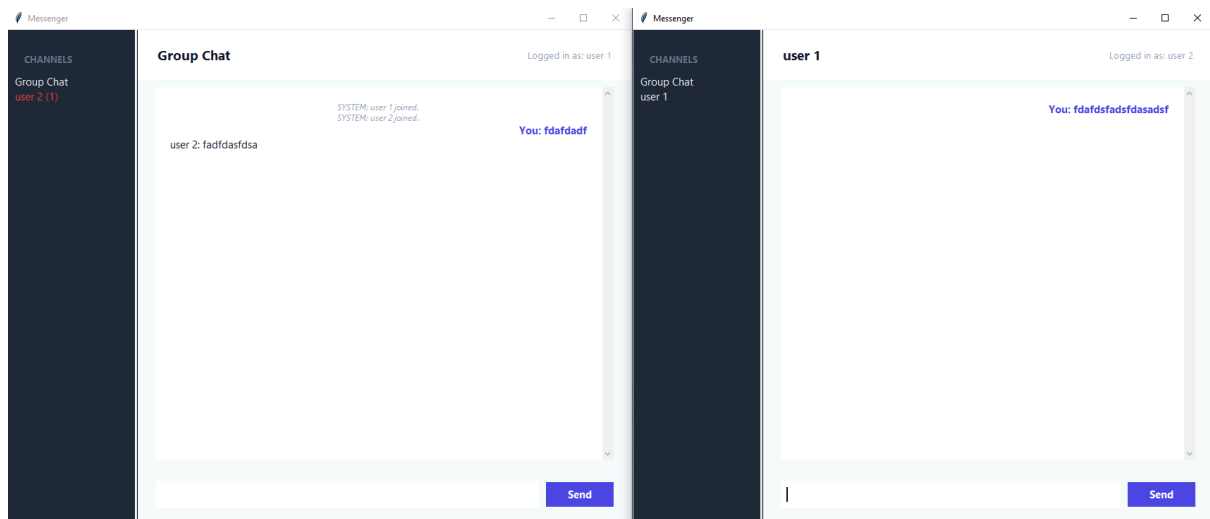
לאחר שהמשתמש בחר את השם המסך יראה כך:



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



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



בתמונה זו רואים שליחה של הודעה פרטית ממשתמש אחד אל השני. כאשר המשתמש השני אינו צפה עדיין בהודעה, השם של המשתמש יראה בצבע אדום.

1. מקביליות וניהול תהליכים (Concurrency)

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

- **בצד השרת:** איך נבנה את לולאת ה-accept כך שהיא לא תחסום את היכולת של השרת לשדר הודעות למשתמשים קיימים?
 - **בצד הלקוח:** מדוע הקוד המקשיב להודעות (receive_messages) חייב לרוץ ב-Thread נפרד מהלולאה הגרפית (mainloop)?
 - **סגירה נקייה:** איך נשתמש בסיגנלים (כמו SIGINT או פקודת EXIT) כדי לסגור את כל ה-Sockets הפתוחים בצורה מסודרת מבלי להשאיר "תהליכי רפאים" במחשב?
- הודעות פרטיות:** איזה "סימן" (כמו הפקודה msg/) נשתיל בתוך המחרוזת כדי שהשרת ידע לנתב את ההודעה למשתמש יחיד ולא לכל הקבוצה?
- עדכונים חיים:** איך נבנה את הודעת ה-USER_LIST כך שהממשק הגרפי ידע לרענן את רשימת אנשי הקשר באופן אוטומטי.

2. ניתוח רכיבי הרשת (Network Traffic)

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

- **לחיצת היד המשולשת (Three-way Handshake):** מופיעה בתחילת כל חיבור חדש (שורות 131-133 ו-233-235).
 - **[SYN]:** הלקוח מבקש להתחבר.
 - **[SYN, ACK]:** השרת מאשר ומבקש לסנכרן.
 - **[ACK]:** הלקוח מאשר סופית. מרגע זה הצינור פתוח.
- **[PSH, ACK]:** סימון (Flag) המעיד על העברת מידע בפועל (Data Transfer). ה-Push אומר למערכת להעביר את המידע מיד לאפליקציה ולא לחכות בבאפר.
- **Length:** גודל המידע בבייטים. זהו המדד 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) מתאים למחרוזת כמו "n\0".
- **שורה 136 (Len=8):** הלקוח שולח לשרת את פרטי ה-Handshake שלו. הגודל מתאים למחרוזת כמו "User 1:0".
- **שורה 138 (Len=17):** השרת שולח את רשימת המשתמשים המעודכנת (USER_LIST:User 1\n).
- **שורה 140 (Len=23):** השרת מפיץ הודעת מערכת: n\SYSTEM: User 1 joined.

שלב ב': התחברות "User 2" (שורות 233-242)

- **שורות 233-235:** המשתמש השני מתחבר (פורט מקור 51900).
- **שורה 236 (Len=2):** השרת שולח ID חדש ("1").
- **שורה 238 (Len=8):** המשתמש השני נרשם כ-"User 2:1".
- **שורה 240 (Destination port 51891):** שים לב! השרת שולח הודעה ללקוח הראשון כדי לעדכן אותו שמישהו חדש הצטרף.

4. הקשר בין הקוד לתעבורה

הלוג מוכיח שהמנגנונים שהטמענו עובדים:

- **הפרדת הודעות (\n):** ניתן לראות שהודעות נשלחות בנפרד (שורות עוקבות עם Length קטן). ה-\n בסוף כל הודעה הוא זה שמאפשר ללקוח לדעת ששורה 138 הסתיימה ושורה 140 היא הודעה חדשה.
- **Broadcasting (שידור):** שורה 240 היא ההוכחה שהשרת יודע לנהל מספר סוקטים בו-זמנית – הוא מקבל מידע מפורט 51900 (User 2) ומעביר אותו לפורט 51891 (User 1).
- **יעילות:** רוב החבילות הן בגודל של 44-70 בייטים. זהו "תקורה" (Overhead) נמוכה מאוד, מה שהופך את הצ'אט למהיר מאוד בזמן אמת.

5. סיכום סטטיסטי של הלוג:

- **סך הכל חבילות:** 50 חבילות TCP.
- **כתובות:** הכל פנימי (1.0.0.127), מה שמעיד על בדיקה מקומית.
- **סטטוס אפליקציה:** הלוג מתעד הצטרפות של שני משתמשים ותחילת סנכרון רשימת משתמשים ביניהם. בנוסף, קיימות רשומות שמייצגות העברת הודעות בין המשתמשים.

6. שימוש ב-AI:

כשכתבנו את הפרוייקט, השתמשנו ב-AI על מנת להמשיך ולחקור לעומק נושאים תוך כדי פיתוח הפרוייקט. אלו חלק מן ה-prompts המייצגים שבעזרתם למדנו ופיתחנו.

שלב 1: הבנת ארכיטקטורת הרשת (Sockets)

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

- **פרומפט להבנת הפרוטוקול:** "מהו תפקידו של SOCK_STREAM ו-AF_INET בקוד השרת? איך הם מבטיחים שההודעות יגיעו ליעדן בצורה אמינה?"
- **פרומפט לפתרון בעיות תקשורת:** "הסבר מדוע הוספנו את המנגנון של תו שורה חדשה (\n) בסוף כל הודעה בשרת וכיצד ה-buffer בקליינט מונע מהודעות 'להידבק' אחת לשנייה."

שלב 2: ריבוי תהליכים ומקביליות (Threading)

הבנה איך השרת מטפל בהרבה אנשים והקליינט מקשיב ומציג בו-זמנית.

- **פרומפט לניהול משתמשים בשרת:** "כיצד השרת משתמש ב-threading.Thread כדי לנהל שיחות עם מספר לקוחות במקביל מבלי שהתוכנה תיעצר?"
- **פרומפט לסגירה מסודרת:** "איך השרת מצליח להקשיב גם לחיבורים חדשים וגם לפקודת 'EXIT' בטרמינל בו-זמנית בעזרת daemon=True?"

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

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

- **פרומפט להפרדת שיחות:** "כיצד אפשר להשתמש במשתנה מסוג מילון כדי להפריד בין הודעות פרטיות לבין ה-Group 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 בפרוטוקול מבוסס web socket?"

שלב 4: למידה מעשית (יצירה מחדש)

בקשות שיעזרו לך לכתוב קוד דומה בעצמך.

- **פרומפט לכתיבת שרת בסיסי:** "הצג לי דוגמא לקוד פייתון בסיסי לשרת TCP שיודע רק לקבל חיבור אחד ולהחזיר 'Hello World' ללקוח, כדי שאבין את הבסיס לפני הוספת ה-GUI."
- **פרומפט ללמידת Tkinter:** "הסבר לי איך משתמשים ב-tag_config בתוך ScrolledText כדי ליישר טקסט לימין או לשמאל, ואיך זה עוזר ליצור UI."

ניהול הודעות: כיצד נבטיח שהודעה לא "תיחתך" או "תידבק" להודעה אחרת בגלל מגבלת ה-MESSAGE_SIZE_IN_BYTES?

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