הפרויקט מכיל שלושה רכיבים עיקריים: שרת (Server), לקוח משדר (TC) ולקוח מקבל (RC).

לכל אחד מהרכיבים יש חלק משמעותי ושונה בפרויקט

**תיאור הארכיטקטורה המוצעת:**

שרת (Server):

השרת אחראי על העברת המידע בין הלקוח המשדר (TC) ללקוח המקבל (RC).

נוסף על כך השרת אחראי על פעולת התמלול והכתיבה למסד הנתונים.

השרת מתחלק לשלושה חלקים: TC\_Handler, RC\_Handler & Transcriptor.

כל אחד מהחלקים אחראי על אחת מהפעולות הראשיות של השרת (קבלה מלקוח משדר, תמלול וסכרון הוידאו והטקסט, העברה ללקוח המקבל).

TC\_Handler:

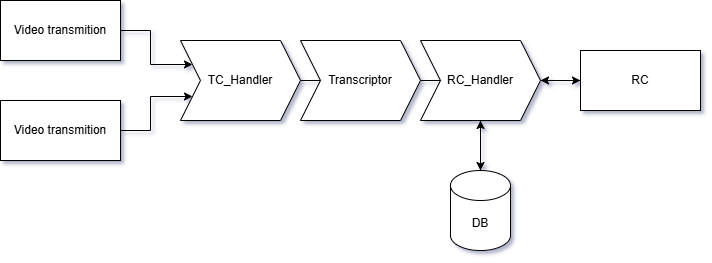
חלק זה בשרת אחראי על קבלת וידאו ואודיו מהלקוח המקבל בעזרת חיבור UDP עם פרוטוקול RTP ומסנכרן ביניהם לפני שהוא מעביר לרכיב השרת הבא.

Transcriptor:

חלק זה לוקח את האודיו ובעזרת קריאת API במטרה לתמלל את האודיו אשר התקבל מה-TC

RC\_Handler:

חלק זה אחראי על שליחת הוידאו והתמלול ל-RC ותקשורת עם ממסד הנתונים בעזרת חיבור TCP, HTTP, RTP ו-WebSockets.



לקוח משדר (TC):

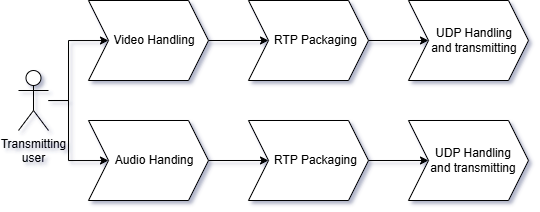
הלקוח המשדר אחראי על שידור הוידאו והאודיו לשרת.

רכיב זה הוא תוכנה לוקלית שמשדרת בעזרת חיבור UDP ופרוטוקול ה-RTP אודיו ווידאו בשני תהליכים נפרדים (פורט לוידאו ופורט לאודיו)

רכיב זה מכיל שני חלקים: Audio\_Handler, Video\_Handler.

כל חלק מהרכיב בנוי משלושה שלבים עיקריים: קליטה - הכנסה לפקטה – שליחה.

לדוגמא: קליטה של וידאו -> הכנסה לפקטת RTP -> שליחה לשרת.



לקוח מקבל (RC):

תפקידו של רכיב ה-RC הוא הצגת שידורי וידאו ותמלול המתקבלים באופן שוטף מהשרת.

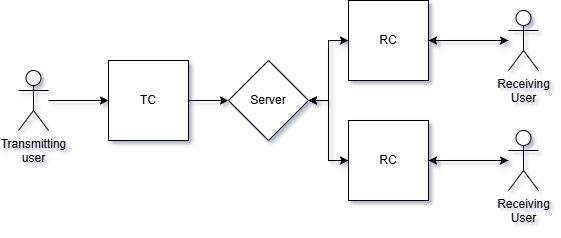
יישום זה מבוסס ווב (Web-based) ומנצל את פרוטוקול WebSocket להעברת נתונים בצורה מהירה ויעילה, המבטיחה תקשורת בזמן אמת (RTC) חיונית לשידורים חיים.

היתרונות בבחירת ארכיטקטורת ווב עבור ה-RC:

* נגישות גבוהה: המערכת נגישה מכל מכשיר המצויד בדפדפן אינטרנט.
* פיתוח ופריסה יעילים: קלות המימוש והתחזוקה בסביבת ווב.
* מתאים למגון רחב של פלטפורמות (cross-platform): עבודה חלקה על פני מערכות הפעלה ודפדפנים שונים ללא צורך בהתאמות ספציפיות.
* חווית משתמש נוחה: ממשק מוכר ואינטואיטיבי המקל על השימוש.

מאחר ומדובר באתר כל הקוד של ה-RC מאוחסן בתוך ה-Server אך ניגשים אל

ה-RC דרך הדפדפן.

כל שלושת הרכיבים הללו ביחד מנהלים את המוצר הסופי והכרחיים לתפעולו היעיל. 

**תיאור הטכנולוגיה הרלוונטית:**

שפות תכנות: השפות השונות המשומשות בפרויקט הזה.

|  |  |  |
| --- | --- | --- |
| שם השפה | שימוש | סיבה לשימוש |
| Python | Server, RC, TC | שפה ראשית של הפרויקט.  נוחה לשימוש, כתיבה מהירה, מכילה ספריות רבות ומאוד ורסטילית. מוכרת במימוש הנוח שלה לרשתות |
| TypeScript | RC | משומשת למימוש ה-RC.  השפה משומשת בצורה נרחבת בעולם פיתוח הווב. נותנת את כל מה שיש ב-JS אבל עם בטחון של OOP ושימוש ב-Types. |
| HTML5 & CSS | RC | שפות ידועות לפיתוח ועיצוב אתרים. |
| SQL | Server | שפה ידועה ובטוחה לכתיבת מסדי נתונים. |

מערכות הפעלה: מערכות ההפעלה המשומשות בפרויקט

|  |  |  |
| --- | --- | --- |
| שם המערכת | שימוש | סיבה לשימוש |
| Windows | Server, RC, TC | מערכת הפעלה ידועה, נוחה לשימוש, ורסטילית ושומשה לפיתוח ובדיקות. |

ניתן להריץ את הפרויקט על כל מערכת הפעלה כל עוד מותקנים התוספים הרלוונטים

פרוטוקולים: הפרוטוקולים המשומשים בפרויקט

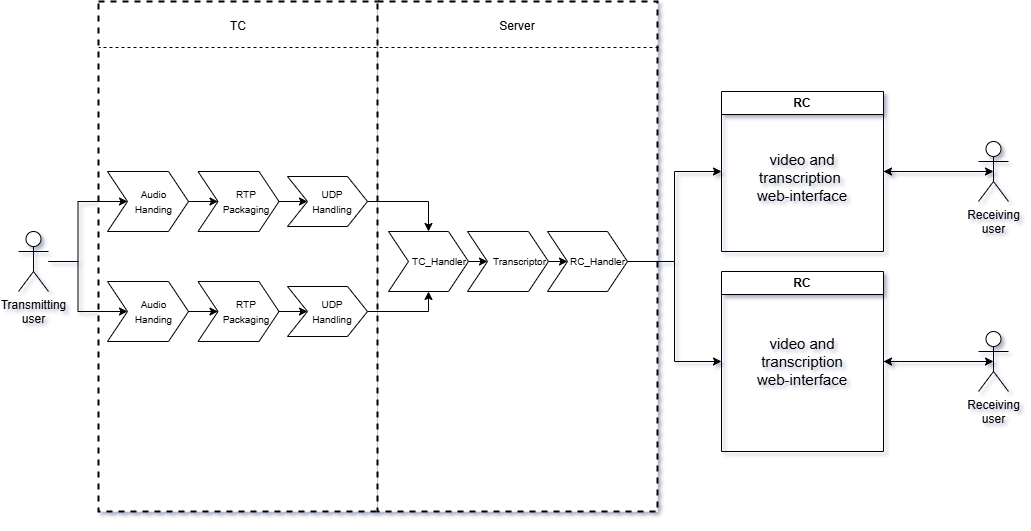
|  |  |  |
| --- | --- | --- |
| שם הפרוטוקול | שימוש | סיבה לשימוש |
| RTP | Server, RC, TC | פרוטוקול ידוע בתעשייה, נוח לשימוש וורסטילי |
| HTTP | Server, RC | פרוטוקול בו משתמשים בשביל לתקשר עם ה-browser |
| Websocket | Server, RC | פרוטוקול מעל HTTP שנותן לשרת יכולת לשלוח לקוח ווב הודעות ללא בקשה של הלקוח. |
| TCP | Server, RC | פרוטוקול אמין, מוודא הגעת הודעות, חובה לשימוש ב-HTTP |
| UDP | Server, TC | פרוטוקול מהיר יותר וידוע בשימוש בשביל סטרימינג |

תחומי עניין רלוונטים:

תקשורת, פרוטוקולים, בינה מלאכותית וטכנולוגיות web.

**תיאור זרימת המידע במערכת:**

להלן תרשים המייצג את התהליך של המערכת



**תיאור האלגוריתמים המרכזיים:**

חלוקת פריימים לפקטות והשלמתם בקבלה חזרה לפריימים:

לאחר קבלת הפריים ראיתי שהיו פריימים שהיו בגודל שהיה שדרש יותר מפקטה אחת. לכן אני החלטתי לבנות אלגוריתם לחלוקת הפריימים לפקטות.

שיטות רעיונות לאלגוריתם זה:

הגדרת מושגים:

* Marker Bit – ביט בפרוטוקול ה-RTP שאפשר להדליק בשביל לסמן בדרים לשרת/לקוח
* Extension – חלק מפרוטוקול הוא ה-extension אשר נותן להוסיף הרחבות לפרוטוקול במידת הצורך.

פתרון #1 – לא נבחר:

לשלוח את הפריים בחלקים, להשאיר לנמען להניח שכל סדרת פקטות שמגיעה בפרק זמן קצר שייכת לאותו פריים.  
פתרון זה עלול לגרום לבעיות כאשר יש איבוד פקטות או חפיפה בין פריימים. בנוסף, ניתן להשתמש בגודל פקטות קבוע מראש ללא התאמה לתוכן, מה שיכול לגרום לבזבוז רוחב פס ולפיצול לא אופטימלי.

פתרון #2 –נבחר:

לחלק את הפריימים לכמות פקטות אשר נרשמת בתוך הפקטות של הפריימים בעזרת הוספה של extension לפרוטוקול ה-RTP.

ה-extension יכיל את כמות הפקטות אשר אליהן חולק הפריים. בפקטה האחרונה אשר תשלח מדליקים את ה-Marker Bit בפרוטוקול.  
  
לאחר קבלת פקטת הmarker בודק השרת אם הוא קיבל את כל הפקטות של הפריים. אם לא הוא ממשיך לחכות אלא אם כן מתחיל להגיע פריים חדש.

במידה ומתחיל להגיע פריים חדש אז השרת זורק את הפריים החסר ומתחיל בקבלה של הפריים החדש.

אני בחרתי בפתרון השני מכיוון שאני הרגשתי שהוא גם הרבה יותר בטוח ויותר מתאים למבנה הפרוטוקול.

**תיאור סביבת הפיתוח:**

סביבות פיתוח (IDE):

במהלך פיתוח פרויקט זה היה שימוש בשתי סביבות פיתוח:

* Pycharm
* Visual Studio Code

כל סביבה שומשה למטרה ספציפית בה היא מוכרת כהכי נוחה ומכילה כלים העוזרים בפיתוח.

|  |  |  |
| --- | --- | --- |
| סביבת הפיתוח | שימוש | סיבת הבחירה |
| Pycharm | פיתוח כל החלקים בפרויקט הדורשים הכתובים ב-Python  (TC & Server) | Pycharm היא סביבת הפיתוח הכי ידועה בשביל Python ומכילה כלים רבים (כגון דיבאגר מצוין) ואינטגרציות רבות (כגון גיט)  נוסף על כך אני משתמש ב-Pycharm שנים רבות ויודע איך לתפעל את התוכנה. |
| Visual Studio Code | פיתוח ה-RC. | VSC היא תוכנת עריכת קוד אשר משומשת רבות לפיתוח של אתרים וכתיבת תוכנה לווב. היא מכילה אינטגרציות של צפייה באתר ב"לייב" בנוסף לעוד כל מיני פיצ'רים נוספים העוזרים בבניית אפליקציות וובץ  התוכנה הזאת היא אחת התוכנות הראשונות שאני כתבתי בהן קוד ואני מכיר אותה היטב, דבר שעזר לי לבחור אותה בשביל הפרויקט הזה |
| ipython | בדיקות וניסיונות ע פייתון | טרמינל פייתון נוח וזמין שנותן פתרון פשוט ומהיר לעשייה של סניפטים של קוד ובדיקות אבל נותן יותר גמישות מ-Python Idle |

מערכת ניהול גרסאות:

בפרויקט זה כמו בכל פרויקט שלי בין אם הוא פרויקט תוכנתי או לא אני השתמשתי ב-git בשביל לשמור את השינויים שלי ולוודא שלא יקרה מצב שמשהו מפסיק לעבוד בפרויקט ואני לא יכול לחזור אחורה.

נוסף על כך על מנת שימוש נוח בגיט וצפייה נוחה בשינויים השתמשתי ב-github התור פלטפורמת ה-git שלי מאחר וזאת הפלטפורמה שאני מכיר הכי טוב.

כלים דרושים לבדיקות:

מאחר ופרויקט זה הוא פרויקט רשתי הכולל את אלמנטים של ווב, פייתון, רשתות, מדיה ועוד כל מיני נדרשים כל מיני כלים אשר עוזרים בבדיקות ודיבאגינג של הקוד והתוצר.

|  |  |  |
| --- | --- | --- |
| כלי | שימוש | סיבת בחירה |
| Wireshark | אבחון תקשורת | כלי מוכר, יש לי ניסיון רב איתו ואני מכיר אותו היטב. |
| Pycharm Debugger | דיבוג סקרפיטים של פייתון | כלי ברמה מאוד גבוהה שנותן תמיכה לדיבוג של תהליכים, תהליכונים רשתות ועוד |
| Chrome Dev Tools | דיבוג ואבחון בדפדפן | הכלי מובנה בתוך הדפדפן ונותן גישה מלאה לתקשורת גם מוצפנת, קוד FE ונותן לערוך שינויים זמניים בשביל בדיקות |
| Python logging | שמירה של נתונים בזמן הריצה שאפשר לנתח אחרי | ספרייה בנויה בפייתון שמספקת את הנדרש בצורה יעילה. |

**תיאור הפרוטוקול:**

בפרויקט יש שימוש בשלושה פרוטוקולים – HTTP, RTP, WebSocket

פרוטוקול ה-HTTP (גרסא מצומצמת):

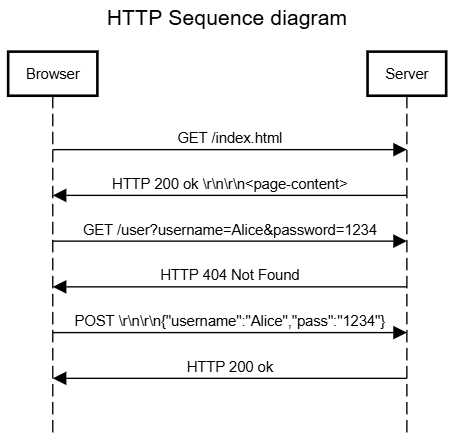
**Hypertext Transfer Protocol (HTTP)** הוא פרוטוקול תקשורת סטנדרטי להעברת מידע בין לקוחות לשרתים ברשת האינטרנט.

הפרוטוקול פועל במודל בקשה-תגובה, שבו הלקוח שולח בקשה לשרת, והשרת מחזיר את המידע המבוקש.

שיטות HTTP עיקריות:

* **GET** - מיועדת לשליפת נתונים מהשרת. המידע נשלח בתוך כתובת ה-URL ואינו משפיע על נתוני השרת.
* **POST** - משמשת לשליחת נתונים לשרת, כגון נתוני טפסים או קובצי מדיה. הנתונים נשלחים בגוף הבקשה, מה שמאפשר אבטחה גבוהה יותר.

הפרדה בין נתונים בבקשותHTTP :

* )? (– מפריד בין נתיבים למשתנים בבקשות GET, לדוגמא: (example.com/page?user=123)
* )&( - מפריד בין שני משתנים שונים (user=123&age=25)
* )=( - מפריד בין שם המשתנה לערכו (name=John).
* \r\n - מסמן מעבר שורה ומפריד בין כותרות הבקשה (Headers)
* \r\n\r\n – הפרדה בין הכותרות לחלק הגוף של הבקשה בו מעבירים את המ ידע המועבר.

מאפייניHTTP :

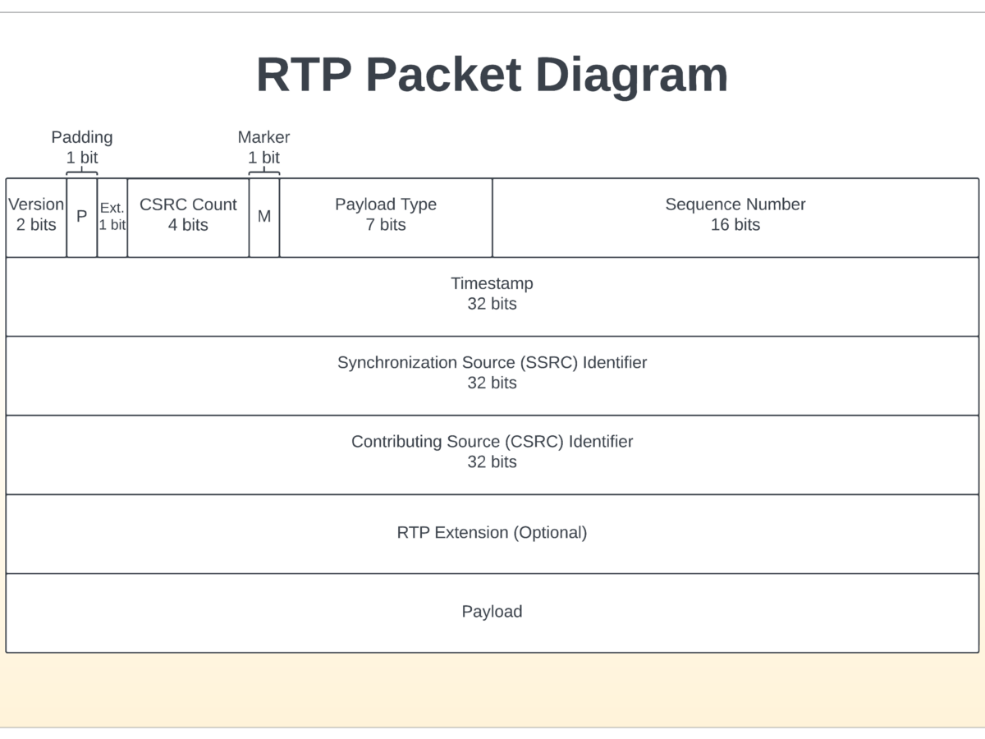
HTTP הוא פרוטוקול stateless, כלומר כל בקשה נשלחת ונענית באופן עצמאי, ללא זיכרון של אינטראקציות קודמות. לשיפור ביצועים ואבטחה.

פרוטוקול ה-RTP:

Real-Time Transport Protocol (RTP) הוא פרוטוקול תקשורת המשמש להעברת מדיה בזמן אמת כגון אודיו ווידאו ברשתות IP. הפרוטוקול פותח כדי לאפשר סנכרון, ניהול רצף נתונים, וזיהוי אובדן פקטות, והוא משמש בעיקר ביישומי סטרימינג ושיחות וידאו.

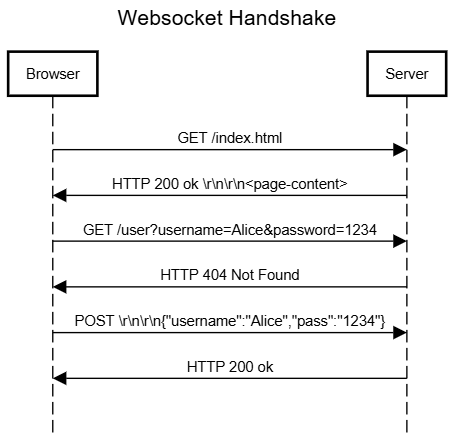
מבנה פרוטוקול ה-RTP:

* **Sequence Number** - מזהה את סדר הפקטות כדי לאפשר שחזור נכון של הנתונים.
* **Timestamp** - מסייע בסנכרון בין זרמי מדיה שונים.
* **Payload Type** - מציין את סוג הנתונים (למשל, קידוד אודיו או וידאו).
* **Synchronization Source (SSRC)** - מזהה את מקור הנתונים כדי להבחין בין זרמים שונים.
* **Contribute Source Identifiers (CSRC)** - מזהה מקורות נוספים של מדיה (למשל, כשכמה משתמשים משתתפים בשיחת ועידה קולית, כל אחד מהם מקבל מזהה ייחודי בתוך הפקטות).
* **Extension Bit (X)** - אם מוגדר ל-1, זה מציין כי קיימים שדות הרחבה נוספים בפקטה. משתמשים בזה כדי להוסיף מידע נוסף מעבר למבנה הבסיסי של RTP
* **Marker Bit (M)** - מסמן פקטה מיוחדת, למשל פקטה שמסמנת את סוף מסגרת וידאו או נקודת התחלה של מקטע חשוב.
* **Payload Type (PT)** - מזהה את סוג הנתונים (לדוגמה, קידוד אודיו/וידאו כמו H.264 או Opus)



פרוטוקול ה-WebSocket:

פרוטוקול הווב סוקט הוא רוטוקול המאפשר תקשורת רציפה מעל תקשורת אינטרנטית לעומת פרוטוקול ה-HTTP שהוא פרוטוקול stateless.

פרוטוקול ה-Websocket מתחיל בלחיצת יד פשוטה דרך פרוטוקול ה-HTTP כמתואר בתרשים בצד.

לאחר לחיצת היד הצדדים המחוברים יכולים להתחיל לשלוח הודעות בעזרת נפרמטרים של הפרוטוקול:

* FIN (1 bit) – מסמן אם זאת הפקטה האחרונה בהודעה.
* RSV1..3 (1 bit each) – ביטים שמורים להרחבות של הפרוטוקול
* Opcode (4 bits) – סוג התוכן (בינארי, טקסט וכ"ו)
* MASK (1 bit) – מציין אם יש מפתח הצפנה בשימוש
* Payload Length (7 bit) – גודל הנתונים המועברים בפקטה. חלק זה יותר מסובך ובנוי בצורה הבאה. במידה ויש צורך בהרחבה אז מתווסף באפר שנקרא (Extended payload length) שהוא תוספת של בתים לאורך הנתונים

|  |  |  |
| --- | --- | --- |
| גודל נתונים | גודל Payload Length | ערך Payload Length |
| עד 125 בתים | 7 ביט | גודל הנתונים |
| 125 - 65,535 בתים | 23 ביט (עוד 2 בתים) | 7 ביט התחלתי = 126  שני בתים נוספים = גודל הנתונים |
| 65,535 ומעלה | 87 ביט (8 בתים נוספים) | 7 ביט התחלתי – 127  8 בתים נוספים = גודל הנתונים |

* Masking Key (4 bytes) – מפתח אקראי המשמש להצפנת הנתונים, **נדרש רק בהודעות מהלקוח לשרת.** לא קיים בהודעות מהשרת ללקוח.
* Payload Data – הנתונים עצמם אשר מועברים בפקטה



**תיאור מסכי המערכת:**

מסכי ה-TC:

מסך פתיחה:

בפתיחת המסך מוצגות שתי אפשרויות – Connect & Conversations

Connect – התחברות לשידור

Conversations – צפייה בשיחות אשר תועדו.

עיצוב:

מסך שידור:

במסך מוצג השידור המתומלל הנשלח מהשרת. במידה ואין שידור תוצג הודעה

עיצוב:

מסך שיחות שתועדו:

במסך מוצגת רשימה של כל השיחות שתועדו – כותרת, ותאריך

לכל שיחה יש שתי אפשרויות, הורדה ומחיקה.

הורדה: הורדה של תמלול השיחה בקובץ טקסט לטלפון

מחיקה: מחיקת תיעוד השיחה מממסד הנתונים.

עיצוב:

מסכי ה-RC:

מסך פתיחה:

במסך זה המשתמש המשדר יוכל לבחור את המצלמה בה הוא רוצה להשתמש וכפתור של התחלת שידור

עיצוב:

מסך שידור:

מסך עם אפשרות המצלמה ואפשרות להפסיק את השידור

עיצוב:

**תיאור מבני הנתונים:**

הפרויקט מכיל ממסד נתונים אחד ופשוט ממסד נתונים של השיחות השונות אשר תועדו וככה הוא נראה:

|  |  |  |  |
| --- | --- | --- | --- |
| Filename: string | Date: string | Title: string | Id: int |
| Conversation-1-23/5/2024.txt | 23/5/2024 | Conversation-1 | 1 |

בטבלה מוצגות כל השיחות השמורות ע"פ מספר מזהה ייחודי, הכותרת, התאריך ושם הקובץ של השיחה.

**סקירת חולשות בפרויקט:**

פרויקט זה אינו מאוד מסובך לתפעול מבחינת אבטחה מאחר שאינו מכיל הרבה דברים.

ממסד נתונים:

כדי להגן מפני פריצות SQL Injection יש שימוש בשאילתות פרמטריות שהן חלק מספריית SQLite בשביל להגן מפני המתקפות הללו. השאילתות בנויות מיישום של (?) בכל מקום בו נועד להכנס מידע ולאחר מכן העברת המידע לפי סדר שלאחר מכן הספרייה בודקת ומאשרת.

Web:

מבחינת הווב הכל משתמש בהצפנת SSL/TLS (Transport Layer Security) ותעודה חתומה עצמאית (ssc) בשביל לשמור על כל המידע מוצפן ומאובטח. בשביל הגנה ממתקפות MITM.

UDP Connection:

בחלק זה גם כן יש שימוש בהצפנת DTLS (Datagram Transport Layer Security). הצפנה זאת חוסמת MITM ומאזינים לא רצויים.