# שער

**Dice**

**מערכת לשמירת קבצים על גבי discord**

**תמונה שמכילה טקסט, סמל, שרטוט, לוגו

התיאור נוצר באופן אוטומטי**

בית ספר:

מגיש: איתמר בר און

ת.ז:

שם המנחה: אלי גולדשטיין

תאריך הגשה:9.6.2024

שם החלופה: סייבר



`

תוכן עניינים

[שער 1](#_Toc168658661)

[מבוא 1](#_Toc168658662)

[מבנה / ארכיטקטורה 1](#_Toc168658663)

[מדריך למשתמש 1](#_Toc168658664)

[מימוש הפרויקט 1](#_Toc168658665)

[רפלקציה 1](#_Toc168658666)

[ביבליוגרפיה 1](#_Toc168658667)

[נספחים 1](#_Toc168658668)

# מבוא

ייזום:

הפרויקט שבחרתי לבצע עוסק בשמירת קבצים על גבי פלטפורמט chat קיימת בשם Discord. הפרויקט נועד למנף (בכלל לא לנצל) את שרתי Discord כפלטפורמה לאחסון נתונים מאובטח וללא עלות. התפקיד העיקרי של שירות זה הוא לאפשר ללקוחות להעלות ולהוריד קבצים בצורה מאובטחת דרך שרת Discord. מרכיבי המפתח של המערכת כוללים לקוח, שרת, מסד נתונים ובוט דיסקורד. זרימת העבודה של הפרויקט נועדה להבטיח שהקבצים מוצפנים, מפוצלים ומאוחסנים באופן ששומר על אבטחתם ושלמותם.

A red line with black text

Description automatically generatedהלקוחות שפרויקט זה פונה אליהם הם אנשים שמחפשים פתרון אחסון חינמי. למשל, אני בטוח שאני לא היחיד שה google drive שלו נראה ככה:

וכנראה שאני לא היחיד שלא שמח לשלם בשביל זה...

וזאת גם הסיבה שבחרתי בפרויקט. גם אני מחפש פתרון חינמי לבעיית האחסון שלי. המוצר המוגמר אמור לפתור את הבעיה הזאת עלידי גישה למקום אחסון בלתי מוגבל בענן של דיסקורד. כשהתחלתי את הפרויקט היו שלושה אתגריים עיקריים שצפיתי:

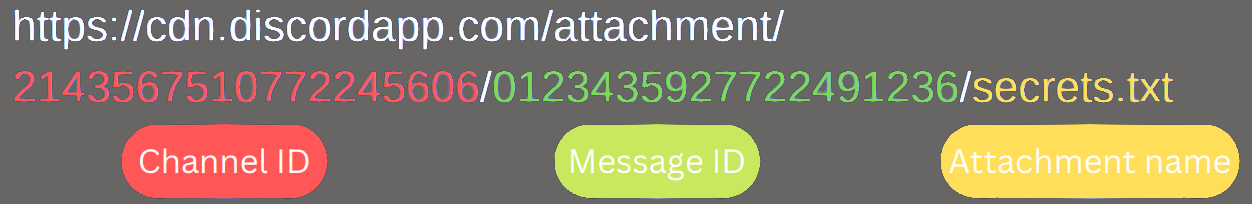
1. הגבלת גודל הקבצים של דיסקורד.
2. פרטיות ואבטחת הקבצים.
3. מציאת הקבצית ושמירה על סדר לאחר העלעתם לענן.

\*לאורך הפרויקט התגלו עוד אתגרים ושינוים לאתגרים קיימים

בואו נדבר עליהם לפי הסדר. ראשית, "הגבלת גודל הקבצים" בעיה זו נובעת מההגבלה של דיסקורד להעלעת קבצים עד 25mb בשביל למנוע מאנשים למלא להם את השרתים בזבל. זאת בעיה עבורנו כי אנחו רוצים לשמור קבצים בכל הגדלים ולא רוצים להיות מוגבלים ל25mb עלובים. (פתרון: פיצול הקבצים לחלקים קטנים ואז להרכיב אותם חזרה, עוד על הפתרון בהמשך)

A screenshot of a computer

Description automatically generated

שנית, "בעית הפרטיות" הבעיה נובעת מכך שכל אחד יכול להוריד מידע מהשרתים של דיסקורד אם יש לו את הקישור המתאים. הקטע הוא שהלינקים של דיסקורד תמיד נמצאים באותו פורמט. אם כמה שקשה "לנחש" את הפרמטרים האלו, זה אפשרי ולמי שיצליח תהיה גישה מלאה לקובץ שלנו. 

\*במהלך הפרויקט דיסקורד הפתיעו, ושנינו את הפורמט. עוד על כך ההמשך.

האתגר השלישי שצפיתי הוא "מציאת הקבצים ושמירה על סדר". כשהתחלתי את הפרויקט היה לי נסיון מעוט מאוד בכתיבת בוטים לדיסקורד. לכן לא ידעתי אם אני אוכל לגשת להודעות שנשלנו, איך שולחים קבצים, אסיפה של הודעות וגישה לאחר זמן. כל אלו היו חדשים ושימשו כאתגר.

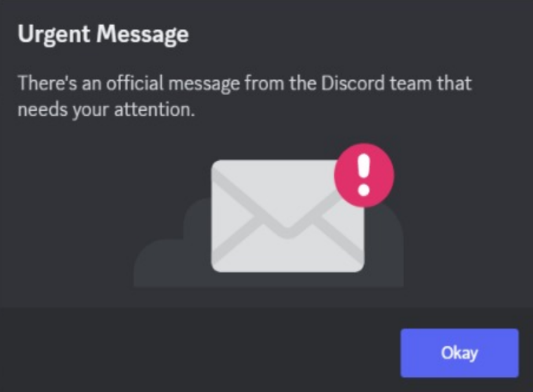
**בעיות תועלת וחסכונות:**

כפי שבטח הבנתם כבר, הבעיה היא שאיחסון קבצים בDrive ובהרבה מקומות אחרים עולה כסף. לכן מה שאנחנו מנסים להשיג הוא פתרון אחסון שלא עולה כסף, discord מאפשרים העלאת קבצים של 25mb ללא הגבלה בכמות, נשתמש בזה כדי להשיג את התועלת שהיא infinite storage 😊. המערכת תיתן שירות שמאפשר להעלות ולהוריד קבצים ללא הגבלה.

**סקירת פתרונות קיימים:**

חשוב להבהיר מה הוא "פתרון" בנושא הזה... פתרון יהיה מערכת שתאפשר העלה והורדה של אינסוף קבצים בחינם לנצח. לכן הייתי אומר שלא קיימים פתרונות כאלו (קונבנציונלים). רוב הספקים יאפשרו לכם נסיון חינמי לתקופת זמן מוגבלת או עבור כמות מקום אחסון מסוים. מה שכן, שמעתי על עוד אנשים שמשתמשים בפלטפורמות כמו YouTube או Minecraft לשמירת קבצים (בצורה לא קונבנציונלית), זאת אחת הסיבות שרציתי לנסות ולעשות את זה על Discord. לאחר שהתחלתי את הפרויקט נודע לי על עוד אנשים שעשו מה שאני עושה (ניתן למצוא קישור בנספחים)

**סקירת טכנולוגית הפרויקט:**

הפרויקט לא עוסק בטכנולוגיה חדשה אלה בגישה לא קונבנציונלית לפתרון בעיה. למרות שהדרך שהפרויקט משתמש בה היא אינה קונבנציונלית היא אכן מוכרת באינטרנט. אחת ההגבלות של הפרויקט היא שהוא תלוי לגמרה בDiscord. בסופו של דבר אנחנו "כאילו" "מנצלים" "חולשה" וברגע שנמאס להם מזה הם יכולים לעצור אותנו ולמנוע מכל הפרויקט לעבוד.

*כמו שאפשר לראות כבר היו כמה רגעים מלחיצים*

**תיחום הפרויקט:**

הפרויקט עוסק בכל התחומים שנדרשו. הפרויקט כולל הצפנות, תקשורת, מסד נתונים ומערכות הפעלה. בפרויקט יש שרת לקוח ומסד נתונים שמתקשרים אחד עם השני על גבי פרוטוקול הצפנה משולב (סימטרי ואסימטרי) , השרת והלקוח משתמשים בThreads, מסד הנתונים נכתב בPython עלידי שימוש בספריה Sqlite3

פירוט תיאור המערכת:

למערכת חמישה חלקים:

1. הלקוח
2. השרת
3. מסד הנתונים
4. בוט הדיסקורד
5. שרת דיסקורד

נדבר עליהם לפי סדר הצגתם, אך לפני כן נגיד כמה מילים על מספר 5, "שרת דיסקורד".דיסקורד נותים אופציה למשתמשים שלהם לפתוח מה שהם קוראים לו "שרת דיסקורד" שרת דיסקורד מקביל לקבוצה בווצאפ. זו דרך ליצור Chat משותף עם כמה אנשים. בשרתי דיסקורד בניגוד לקבוצות ווצאפ יכול להיות יותר מChat אחד ויכולות להיות גם שיחות (או בשפה של דיסקורד Voice Chats) לא מדובר ברכיב שאני תכנתתי אלה זה פשוט מה שאנחנו נקרא לו "דיסקורד" מעכשיו, מה שנמצא שם נמצא אצל דיסקורד ונחשב כשמור בענן.

**הלקוח**

השירות שהלקוח מקבל הוא האופציה לאחסן קבצים על שרת הדיסקורד. ללקוח כמה יכולות:

הרשמה/התחברות – כל לקוח יכול ליצור או להתחבר למשתמש בעזרת שם משתמש וסיסמה.

העלאת קובץ – כל לקוח יכול להעלות קבצים בכל גודל מהמחשב שלו לאחסון על שרת הדיסקורד.

הורדת קובץ – כל לקוח יכול להוריד קבצים שהוא עצמו העלה, מדיסקורד למחשב שלו.

מחיקת קובץ – כל לקוח יכול למחוק קבצים שהוא העלה לשרת הדיסקורד.

**השרת**

השרת משמש כ"מנהל משימות" של כל המערכת הזאת. הלקוחות שולחים לו את הבקשות והוא מטפל בהם, כשצריך הוא יודע להפנות אותם את גורמים אחרים כמו מסד הנתונים ובוט הדיסקורד.

אימות והקצאה - השרת אחראי על אימות של כל הנתונים בין הלקוח לבין שאאר הגורמים למשל נתוני ההתחברות, שמות קבצים וכו... בנוסף השרת אחראי על הקצאת מקום עבור נתונים חדשים. למשל אם משתמש חדש מנסה להירשם או שקובץ עם אותו שם הועלה כבר.

הצפנת קבצים –השרת מצפין את כל הקבצים הנכנסים ומפענח את כל הקבצים היוצאים.

פיצול הקבצים – השרת אחראי על פיצול הקובץ לחלקים וגם על הרכבתו בין המעבר מלקוח לדיסקורד או דיסקורד ללקוח.

ניהול הקבצים – השרת מקטלג את הקבצים של כל לקוח מול מסד הנתונים בשביל לוודא שניתן להשיג את הקבצים חזרה בצורה בטוחה לאחר העלעתם.

**מסד הנתונים**

מסד הנתונים שומר מעט מאוד דברים אך אלו דברים קריטים, הוא שומר מידע על הלקוחות ועל הקבצים שלהם.

נתוני התחברות – מסד הנתונים שומר נתוני התחברות של משתמשים (שם משתמש וססמה).

נתוני הקבצים – מסד הנתונים שומר מידע על איפה מאוחסן כל קובץ, לאיזה משתמש הוא שייך ואיך להגיע אליו.

**בוט הדיסקורד**

בוט הדיסקורד אחראי על התקשורת בין השרת לדיסקורד, הוא משתמש ב Discord API בשביל לאפשר לו לעשות דברים כמו לשלוח הודעות, להגיב על הודעות, ליצור Channels חדשים וכו (פרטים מעמיקים יותר בהמשך)

העברת הקבצים – הבוט מקבל את חתיכות הקבצים ואת הוראות השליחה מהשרת ולפי הן שולח את הקבצים בדיסקורד.

הקצאת מקום - כאשר מתחבר לקוח חדש השרת מודיע לבוט והוא בהתאם יוצר לו מקום חדש עבור הקבצים שלו.

מעקב אחר הודעות – הבוט יודע למחוק הודעות (למחוק קבצים) לבדוק הודעות ישנות לפי צורך ולהגיב בהתאם.

**שרת הדיסקורד**

כפי שהתחלתי להסביר בהתחלה, לא מדובר ברכיב שאני תכנתי אלא הדרך שלנו לדבר עם **ה**שרתים של דיסקורד. בשרת זה יש צ'אטים (בשפה של דיסקורד channels ) בשרת כל לקוח שלנו מקבל channel שבו הבוט שולח את הקבצים ששיכים לו.

ערוצי אחסון (storage channels) – אנחנו משתמשים בצ'אנלים בשרת כ"תאי אחסון" עבור כל

לקוח.

**פירוט בדיקות:**

בגדול עשיתי בדיקות רצניות (stress test) רק כאשר סיימתי לעבוד על שלבים משמעותיים בפרויקט. הנה כמה דוגמאות:  
העלאה והורדה קובץ גדול : לאחר שסיימתי את המנגנון שמטפל בפירוק הקבצים עשיתי בדיקה שבה העלתי קובץ של 1MB הקובץ עלה וירד כמתוכנן.

מחיקת קובץ – מחקתי קובץ שהיה שמור והקובץ נמחק גם מהדיסקורד וגם ממסד הנתונים.

הרשמה והתחברות – הרשמה והתחברות מהקליינט.

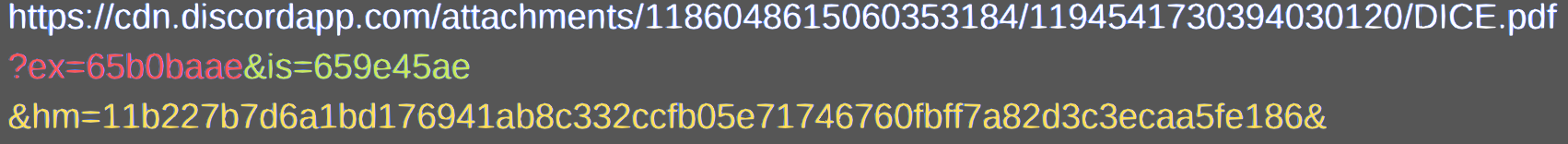
כמה לקוחות בו זמנית – ערכתי בדיקה בה כמה לקוחות מתחברים ומבצעים פעולות בו זמנית.

בדיקות הצפנה – ערכתי הרבה בדיקות להצפנה, לא בגלל שהיא רכיב מפתח אלא כי התקשתי לגרום לה לעבוד כמו שרציתי.

**לוז הפרויקט:**

עבור לוז הפרויקט ניתן לראות את "יומן שיקוף" בנספחים.

**ניהול סיכונים:**

הסכנה העיקרית בפרויקט הזה היא שהבכל רגע נתון דיסקורד יכולים לשנות משהו שיהפוך את הפרויקט לבלתי שמיש. היו מקרים שבהם הם עשו שינוי שכמעט גרם לכך. למשל כאשר הם ביצעו שינוים לדרך שבה הם שומרים דברים על השרתים שלהם. מהפורמט הישן שאותו הראתי לכם בהתחלה הם שינו לפורמט חדש שנראה כך:

כפי שניתן לראות הפרמטרים הקודמים (channel ID, Message ID, Attachment name) נשארו אותו דבר אך הווספו שלושה פרמטרים חדשים ( ?ex, &is &hm) אלו בעלי התפקידים הבאים:

?ex – קובע מתי הלינק פג תוקף.

&is – קובע מתי הלינק נהיה פעיל.

&hm – checksum כלשהו.

למזלנו, דבר זה לא השפיע עלינו בגלל שהלינקים שאנחנו משתמשים בהם לא נשמרים מחוץ לדיסקורד. אלה נלקחים ישירות מדיסקורד ברגע שאנחנו זקוקים להם, ולכן אנחנו נקבל לינק מחודש ככה שהוא תמיד יהיה בתוקף.

תיאור תחום הידע - פרק מילולי:

**צד הלקוח**

**שם הפעולה:** החברות/הרשמה.

**מהות:** הכנסת המשתמש למערכת / רישום משתמש למערכת – קליטת פרטים אישיים

**אוסף היכולות הנדרשות:**

* ממשק משתמש
* קליטת נתונים
* בדיקת תקינות
* שליחה לשרת
* בדיקת תקינות מול מסד הנתונים
* קבלת תושובה מהשרת
* הצגה למשתמש
* ממשק משתמש

**אובייקטים נחוצים:** ממשק משתמש, תקשורת, מסד נתונים

**שם הפעולה:** העלאת קובץ.

**מהות:** העלעת הקובץ לשרת ושליחתו בדיסקורד

**אוסף היכולות הנדרשות:**

* ממשק משתמש
* קליטת נתונים
* בדיקת תקינות
* הצפנה
* שליחה לשרת
* פיצול הקובץ אצל השרת
* העברה לבוט
* שליחה בדיסקורד
* שמירת נתונים במסד הנתונים
* קבלת תשובות מהשרת
* הצגה למשתמש

**אובייקטים נחוצים:** ממשק משתמש, תקשורת, בוט דיסקורד, מסד נתונים

**שם הפעולה:** הורדת קובץ.

**מהות:** להרכיב את הקובץ ולהוריד אותו

**אוסף היכולות הנדרשות:**

* ממשק משתמש
* שליחת בקשה לשרת
* אסיפת נתונים ממסד הנתונים
* הבוט אוסף את החלקים
* הרכבה של החלקים בשרת
* פיענוח
* קבלת תשובות מהשרת
* הצגה למשתמש

**אובייקטים נחוצים:** ממשק משתמש, תקשורת, בוט דיסקורד, מסד נתונים

**שם הפעולה:** מחיקת קובץ.

**מהות:** למחוק את הקובץ מהדיסקורד ולנקות אותו ממסד הנתונים

**אוסף היכולות הנדרשות:**

* ממשק משתמש
* שליחת בקשה לשרת
* אסיפת נתונים ממסד הנתונים
* הבוט מוחק את החלקים
* מחיקה מהמסד בשרת
* קבלת תשובות מהשרת
* הצגה למשתמש

**אובייקטים נחוצים:** ממשק משתמש, תקשורת, בוט דיסקורד, מסד נתונים

**צד השרת**

**שם הפעולה:** הקצאת מקום חדש לשמירת קבצים

**מהות:** יצירת ערוץ חדש בשביל שמירת מידע

**אוסף היכולות הנדרשות:**

* עדכון מסד הנתונים
* בקשה מהבוט ליצירת ערוץ
* עדכון השרת
* עדכון המסד
* עדכון הלקוח

**אובייקטים נחוצים:** בוט דיסקורד

**שם הפעולה:** פיצול הקבצים

**מהות:** פיצול קבצים גדולים לחתיכות של 25mb

**אוסף היכולות הנדרשות:**

* פירוק הקובץ

**אובייקטים נחוצים:** השרת

**שם הפעולה:** הצפנת קבצים (פיענוח אותו דבר)

**מהות:** הצפנת הקבצים לפני שמירתם

**אוסף היכולות הנדרשות:**

* בקשת הקבצים מהבוט
* חיבור הקבצים
* פיענוח הקבצים
* שליחה ללקוח

**או**

* קבלת הקבצים מהלקוח
* הצפנה
* פירוק לחלקים
* שליחה לבוט

**אובייקטים נחוצים:** בוט דיסקורד

**שם הפעולה:** ניהול קבצים

**מהות:** שמירת הפרטים של הקבצים (משמשים כמצביעים)

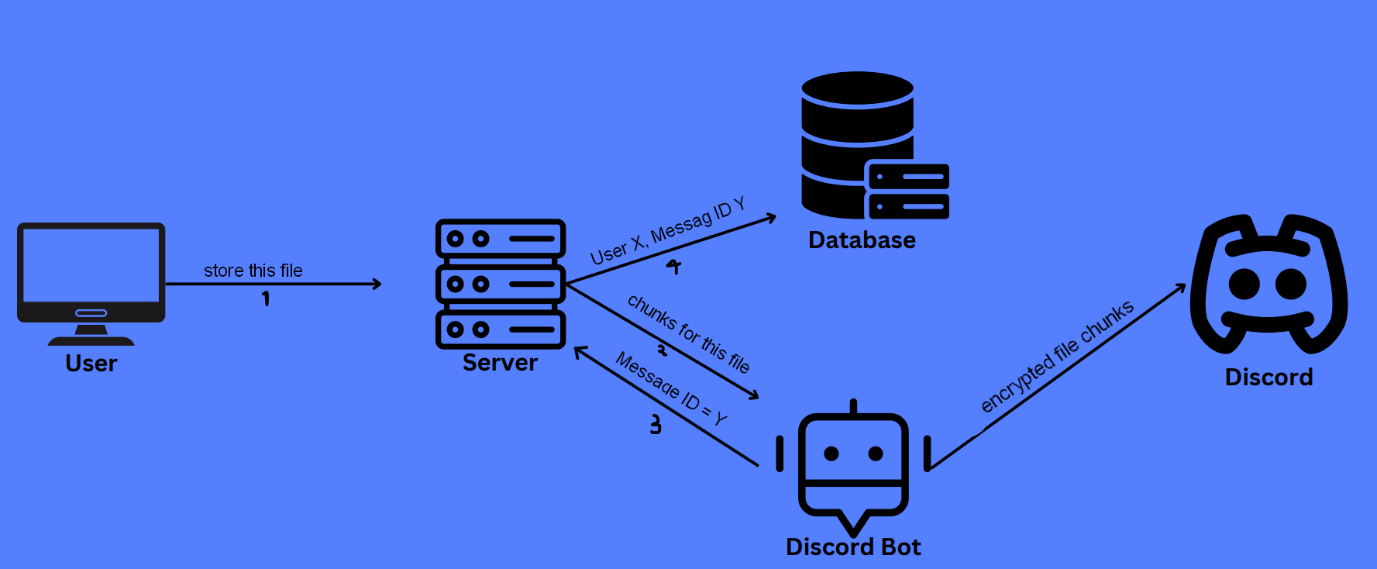
**אוסף היכולות הנדרשות:**

* מידע מהבוט
* עיבוד המידע
* שמירה במסד הנתונים

**אובייקטים נחוצים:** בוט דיסקורד, מסד נתונים

# מבנה / ארכיטקטורה

**תיאור הארכיטקטורה של המערכת:**

****כמו שכבר ראינו, המערכת מורכבת מחמישה רכיבים עקריים. לקוח, שרת, בוט, מסד נתונים ושרת דיסקורד. בשביל להבין טוב יותר את הקשר בניהם בואו נראה כמה דוגמאות לתקשורת בניהם:  
דוגמה להעלעת קובץ:

A diagram of a server

Description automatically generatedדוגמה להורדת קובץ

כמו שניתן לראות בתרשימים. הסרבר משמש כHub עבור כל הישויות (חוץ מדיסקורד ישירות) . תקשורת בין מסד הנתונים, הלקוח והבוט קוראת דרך השרת. עוד ניתן לראות שהתקשורת עם דיסקורד מתנהלת דרך הבוט בלבד והוא מנוהל מהשרת.

**תיאור טכנולוגיות רלוונטיות:**

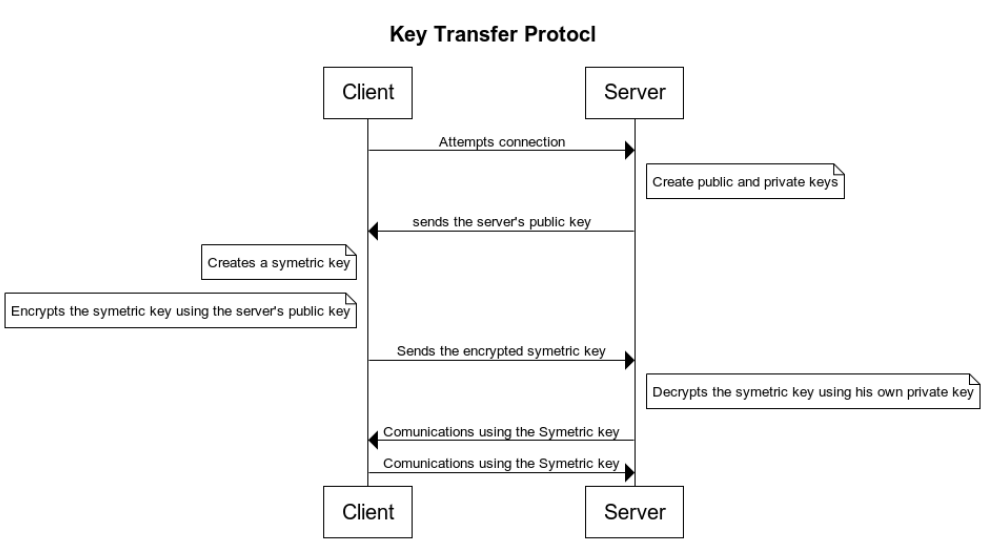
* שפת תכנות: Python.
* מערכת הפעלה: Win.
* תקשורת: שרת מרכזי שמדבר עם הרבה רכיבים + בוט דיסקורד כדי לתקשר עם דיסקורד.
* תחום עניין: שמירת קבצים בדרך יצירתית.

**אלגוריתמים מרכזיים בפרויקט**

בפרויקט קיימים כמה אלגוריתמים מרכזיים. בניהם: ניהול לקוחות בצד השרת כולל העברת מפתחות, ניהול מסד הנתונים, תהליך שמירת הקבצים.

תחילה נדבר על **ניהול הלקוחות בצד השרת**. עלמנת לאפשר לשרת לנהל כמה לקוחות בו זמנית אנחנו עושים משהו כזה: ראשית השרת מאזין לחיבורים, כאשר הוא תופס חיבור הוא מוסיף אותו למילון באופן הבא {client\_id : [socket\_object, database]} כך שclient id הוא מזהה יחודי (מספר) עבור הלקוח הזה וה socket object הוא הסוקט של הלקוח המחובר כך שאם נרצה לשלוח לו משהו נשתמש באוביק הזה. לגבי ה,database זה בגלל שאנחנו עובדים עם sqlite3 אז צריך ליצור את זה עבור כל לקוח (זה לא חשוב לתהליך ניהול הלקוחות ולכן לא אכנס לזה לעומק פה) לאחר שהוספנו את החיבור למילון הלקוחות שלנו השרת מרים Thread עבור הלקוח החדש ושם מתנהל כל התקשורת בין הלקוח לשרת. בנוסף לכך הדבר הראשון שקורה על הThread הוא תהליך החלפת מפתחות. השרת שולח מפתח פומבי ללקוח שמקבל אותו ויוצר מפתח סימטרי, הלקוח מצפין את המפתח הסימטרי עם המפתח הפומבי ושולח אותו לשרת, השרת מפענך עם המפתח הפרטי ואז כל התקשורת ממשיכה עם המפתח הסימטרי.

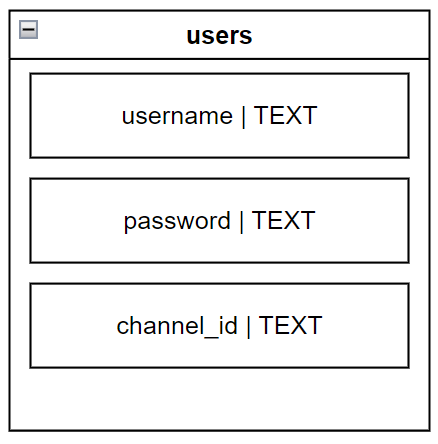
\*(ניתן לראות תרשים זרימה של אלגוריטם זה בעמוד הבא)

**תרשים זרימה – העברת המפתחות**

**תרשים זרימה – ניהול לקוחות בצד השרתA diagram of a company

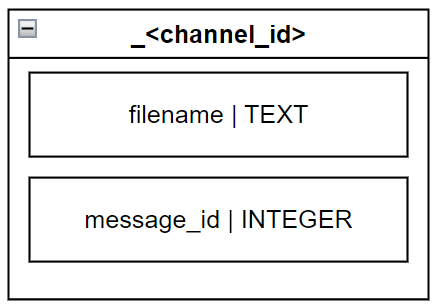
Description automatically generated with medium confidence**

שנית נדבר על **ניהול מסד הנתונים**, אני משתמש בSqlite3 בשביל מסד הנתונים. מסד הנתונים שלנו הוא מסד טבלאי. בואו נדבר על איך הוא מאורגן, במסד תמיד תוכלו למצוא את טבלת ה’users’ טבלה זו אחראית על שמירת פרטי ההתחברות של המשתמש והיא תיראה כך:

username – שם משתמש, נקבע ע"י הלקוח והינו מסוג טקסט.

password – ססמה, נקבעת ע"י הלקוח והינו מסוג טקסט.

channel\_id – מזהה ערוץ, נקבע ע"י השרת מכיל מספרים אך גם הוא מסוג טקסט. (תכף נבין מה הסיבה לכך)

בנוסף לטבלת ה’users’ לאחר שהתחבר לקוח אחד לפחות נוכל למצוא עוד טבלאות, עבור כל לקוח. נקרא להן `\_<channel\_id>` (אך במסד הנתונים הם נראים יותר דומה ל ‘\_1248256729’) הטבלאות האלו הן טבלאות שנפתחות עבור כל לקוח באופן אוטומתי והן שומרות את כל הקבצים של אותו לקוח (אשר שמורים בchannel\_id המתאים) הן גם הסיבה שchannel\_id ב ‘users’ הוא סוג טקסט. בנוסף הסיבה לכך שיש "\_" בתחילת שם הטבלה היא שsqlite3 לא מאפשר ששם של טבלה יתחיל במספר לכך הוספה של התוו "\_" פותרת את הבעיה. הטבלה נראת כך:

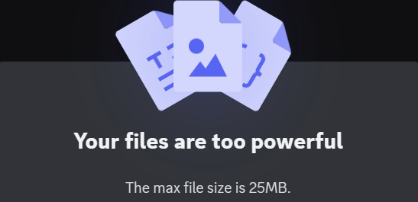
Filename – שם הקובץ, נקבע לפי מה שהלקוח העלה, הינו מסוג טקסט ומשמש בעיקר להצגת שם הקובץ עבור הלקוח.

Message\_id – מזהה ההודעה שמקושרת לקובץ, נקבע לפי בוט הדיסקורד (דרך השרת כמובן) והינו מסוג שלם. משמש בשביל להרכיב את הקובץ חזרה מדיסקורד.

כל ההתנהלות מול ה database היא דרך המחלקה 'DatabaseManager' אשר מנוהלת בשרת.

**תהליך שמירת הקבצים**

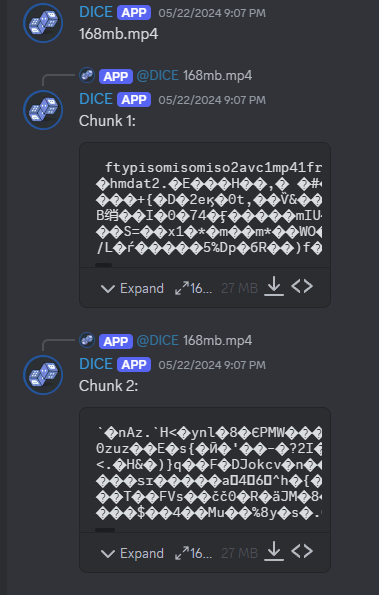
A blue line drawing of a document

Description automatically generatedאלגוריטם זה הוא אלגוריטם מרכזי בפרויקט והוא בין היתר פותר בעיה שהייתה איתנו מתחילת הפרוטיקט. הבעיה היא שדיסקורד מגבילים גודל ההעלה של קבצים לפלטפורמה שלהם. ולכן תיאורתית נוכל לשמור קבצים עד 25mb עלובים. אנחנו לא רוצים את זה, ולכן מצאנו פתרון יצירתי, הרי קובץ בסופו של דבר הוא מידע בינארי, אז מה זה משנה אם הוא נשמר ביחד או בנפרד? כל עוד אנחנו יודעים איך לחבר אותו אחר כך לא אמורה להיות לנו שום בעיה. וזה מה שהפתורון מציע!

על ידי פיצול הקובץ לקבצים קטנים יותר נוכל לשמור אותו על גבי דיסקורד רק חשוב לזכור איך להרכיב אותו חזרה.

עכשיו שאנחנו יודעים איך אנחנו פותרים את בעית הגודל על הנייר בואו נבין איך אנחנו עושים זאת בפועל מול דיסקורד. כמה דברים חשובים שיש להכיר לפני. בשרת דיסקורד שהוא שכמו שאמרנו קודם הוא מקביל לקבוצות ווצאפ יש "ערוצים" ערוצים האלו הם כמו צאט וכל אחד כזה הוא standalone מה שנמצא בערוץ1 לא יופיע בערוץ2. לכל ערוץ כזה יש מזהה יחודי שנקרא לו channel\_id. עוד דבר שיש לדעת הוא שלכל הודעה יש גם מזהה יחודי והוא נקרא message\_id שני המזהים האלו הם קבועים ולא משתנים. למען ההסבר נשתמש במוסגים אלו.

\*חשוב לציין שההסבר הזה מסביר מה קורה אחרי שהקובץ מגיע מהלקוח אל השרת ועובר הצפנה.

לאחר שהקובץ הוצפן השרת במקש מהמחלקה `FileManager` לפצל אותו לחלקים. המחלקה מחזירה את החלקים והם עוברים את בוט הדיסקורד ביחד עם שם הקובץ. הבוט בודק את זמינות שם הקובץ (האם קובץ עם שם זהה הועלה כבר) אם השם אינו זמין הבוט יוסיף (n) אחרי שם הקובץ (n – מספר שמסמל כמות כפלויות). לאחר שהבוט מחליט על שם הקובץ הוא שולח בטקסט את שם הקובץ כהודעה בchannel שמתאים לchannel\_id של הלקוח שביקש את העלעת הקובץ. לאחר ששלח את ההודעה (שם הקובץ) הוא שומר את הmessage\_id שלה בשביל שנוכל לחזור אליה אחר כך ומתחיל לשלוח את החתיכות של הקובץ כ “replays” להודעה המקורית (שם הקובץ)

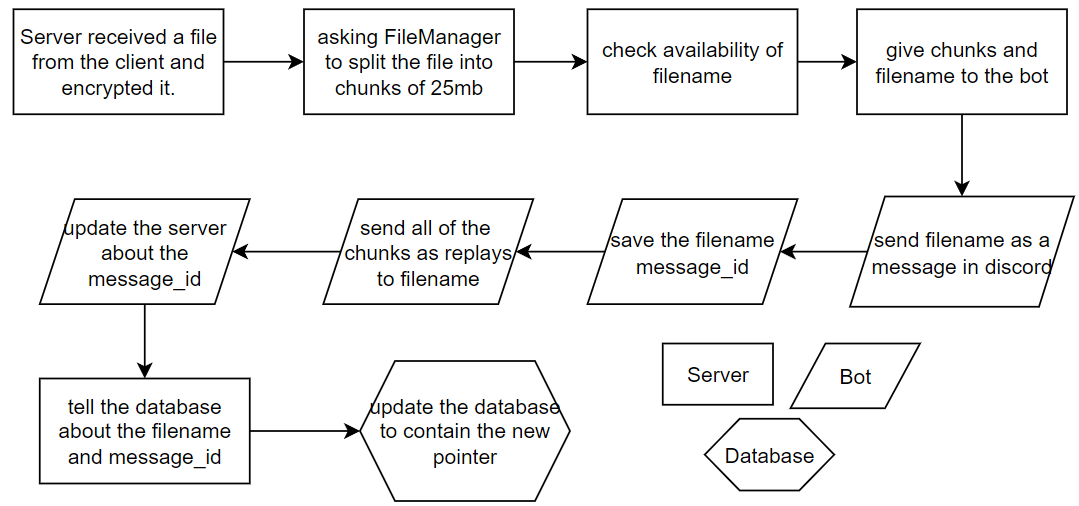
בצד של דיסקורד זה נראה ככה:  
בדוגמה זו ניתן לראות שנשלח קובץ בשם “168mb.mp4”

לקובץ אין כפילויות כי אין (n) לאחר השם. ניתן לראות גם את שירשור

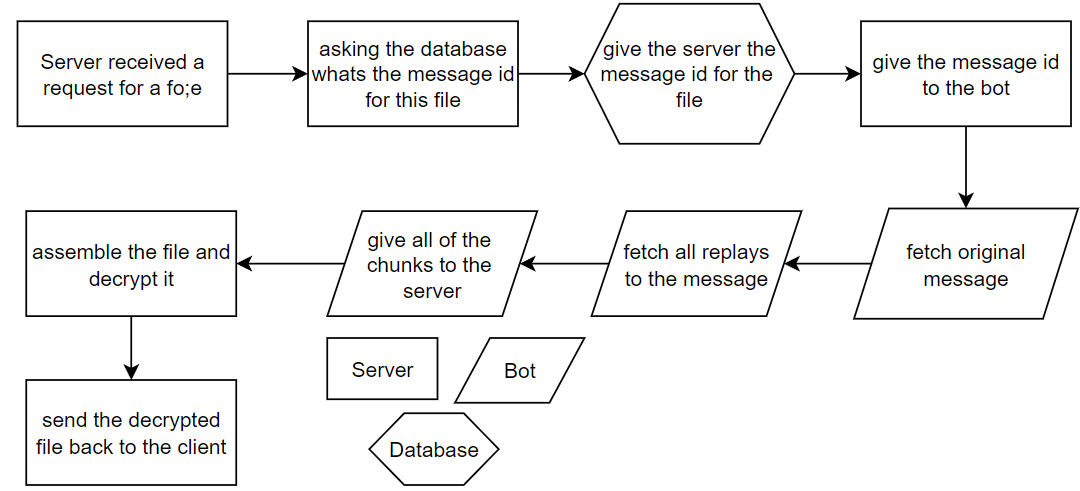
חלקי הקובץ אשר נשלחים כreplays (רואים רק חלק כמובן)

עכשיו שהבנו איך הדברים נשלחים בדיסקורד קל מאוד להבין אין אנחנו מרכיבים אותם חזרה.

כאשר מגיע בקשה לקובץ מסוים השרת מתשאל את מסד הנתונים עבור ה message\_id של ההודעה המקורית, הבוט מוצא את ההודעה המקורית ולוקח את כל הreplays שלה, הוא מעביר אותן לסרבר והסרבר מחבר ומפענך אותם. לבסוף הקובץ חוזר אל הלקוח שלם.

**תרשים זרימה – שליחת קובץ**  


**תרשים זרימה – הרכבת קובץ**

****

**תיאור סביבת הפיתוח:**

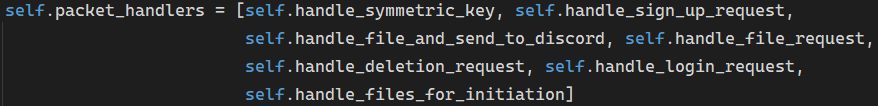
הכלים שהשתמשתי בהם הם: Pycharm, Discord וגם DB Browser(Sqlite)שאינו היה חובה אך עזר לי רבות לבדיקות שקשורות למסד הנתונים. לעומת זאת Discord הוא כלי חשוב למען הבדיקה ולכן ארחיב עליו עוד במדריך למשתמש.

**תיאור פרוטוקול התקשורת:**

הפרוטוקול שלי הוא פרוטוקול יחסית פשוט ועוד על TCP. דבר ראשון בואו נראה את מבנה הפקטה:

ככה תראה פקטה שמעבירה מידע שהוא לא קשור להעברת קבצים (למשל התחברות)

ככה תראה פקטה שמעבירה קובץ בין הלקוח והשרת

בואו נתחיל לדבר על הדברים שזהים בשניהם. תחילה packet id זהו מזהה יחודי עבור כל פקטה ואורכו מספר אחד. המהות שלו היא לאפשר טיפול יעיל בפקטה. איך הוא עושה זאת? שמח ששאלתם. גם בשרת וגם בלקוח נמצת רשימה בשם packet\_handlers שנראת כך: 

כאשר הלקוח או השרת מקבלים פקטה הם מסתכלים על הpacket id ובעזרתו יודעים לאיזה אחת מהפעולות לקרוא. עכשיו מה לגבי len? Len הוא אורך המידע בבפקטה (במקרה של הקובץ האורך של info) אורך שדה זה הוא 4 מספרים (יכול להיות גם 0021 למשל) והוא הפרמטר שהפעולות בpacket\_handlers לוקחות. לפיו הפעולה יודעת כמה לקרוא מהסוקט.

הפרמטר הבא הוא יחודי רק לקובץ והוא “info” הוא מכיך פרטים על קובץ את שם הקובץ וגודל הקובץ. כאשר הצד המקבל, מקבל פקטה כזו הוא קורא את המידע על הקובץ בעזרת “len” ואז קורא את “data” שהיא תכולת הקובץ בעזרת size שנמצא בתוך “info”.

A black screen with white text

Description automatically generatedאחרון חביב הוא “data” בפקטה מסוג קובץ הוא יכיל את תכולת הקובץ אך בפקטה רגילה זה מעט שונה. הוא אך המידע שאנחנו רוצים להעביר אך הוא יהיה מסודר בצורת מילון ויעבור תהליך סיריליזציה בעזרת JSON ורק אז ישלח בפקטה. לדוגמה:

פקטה זו שאחראית על התחברות המשתמש ומעבירה שם משתמש וססמה תעביר אותם באופן הזה. בתוך ממילון.

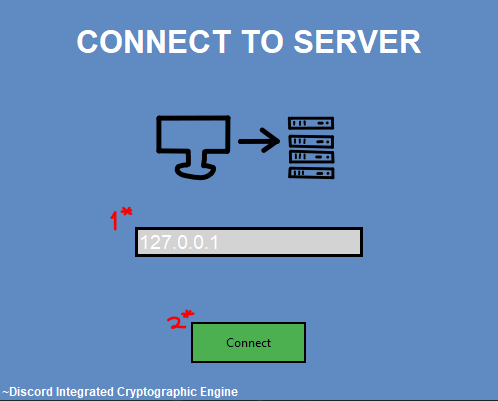
חשוב לציין שהמידע שעובר בתקשורת (חוץ מהמפתחות)

מוצפן בהצפנה סמטרית ככה שבפועל פקטה תיראה ככה:  
<80026b’12\xa17\54\2bc42\2652\fex\’>

כאן packet id = 8, len = 26, והמידע מוצפן אך הוא יהיה מילון שמכיל מידע.

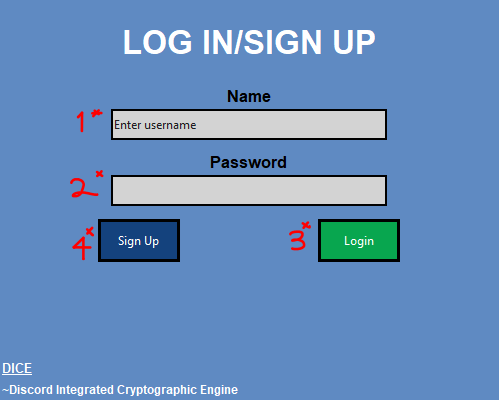
**תיאור מסכי המערכת**

במערכת שלושה מסכים עיקריים. התחברות לשרת, התחברות למערכת, מסך ראשי.

נעבור עליהם לפי סדר הופעתם בפרויקט. תחילה נדבר על מסך ה**התחברות לשרת**. מסך זה הוא בסיסי מאוד וכל התפקיד שלו הוא התחברות לשרת. במסך יש שדה טקסט אחד שבו המשתמש נדרש להכניס כתובת IP שאליה הוא רוצה להתחבר כלקוח (כתובת הIP של השרת) . בנוסף לשדה הטקסט יש כפתור בצבע ירוק עם הכיתוב "connect" הוא אחראי על החיבור לשרת.

\*1 – שדה הכנסת הטקסט: כברירת מחדל מוצגת הכתובת הלוקאלית אך ניתן לשנות אותה לפי הצורך.

\*2 – כפתור ההתחברות: בליצה עליו יתבצע נסיון התחברות לשרת עם כתובת הIP שהוכנסה בשדה הטקסט למעלה (1)

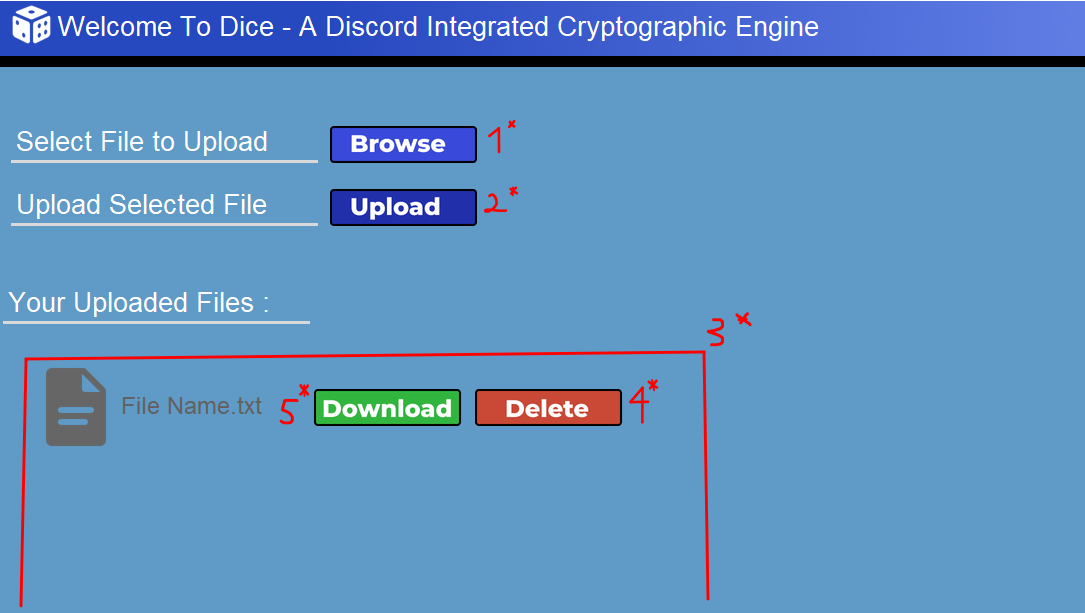
המסך השני שעליו נדבר הוא **מסך ההתחברות למערכת**. מסך זה מציג למשתמש שתי שדות טקסט ו2 כפתורים. בשדה העליון מתבקש המשתמש להכניס שם משתמש ובשדה התחתון הוא מתבקש להכניס סיסמה. הכפתור הימני (הירוק) אשר כתוב עליו log in משמש להתחברות, בעוד שהכפתור השמאלי (הכחול) שכתוב עליו sign up משמש להרשמה.

\*1 - שדה השם: יש להכניס שם משתמש שנמצא במערכת אם אתם רוצים להתחבר או שם שלא נמצא אם אתם רוצים להרשם

\*2 – שדה הססמה: יש להכניס סיסמה מתאימה למשתמש או חדשה אם אתם נרשמים.

\*3 – כפתור ההתחברות: לאחר מילוי כל השדות לחיצה על כפתור זה תנסה להתחבר למשתמש לפי הנתונים

\*4 – כפתור ההרשמה: לאחר מילוי כל השדות לחיצה על כפתור זה תנסה ליצור משתמש חדש ואם תצליח אז היא גם תתחבר אוטומתית למשתמש החדש.

אחרון אבל הכי חשוב, **המסך הראשי.** במסך זה מוצגות האופציות לבחור קבצים לשליחה, לשלוח את הקבצים, לראות את הקבצים השמורים שלך, להוריד קבצים ולמחוק קבצים.

\*1 – כפתור הBrowse פותח חלון לבחירת קבצים (החלון הוא חלון דיפולטי של ווינדוז ולכן אין צורך בהרחבה עליו)

\*2 – כפתור ה upload מאפשר ללקוח לשלוח את הקובץ שנבחר מוקדם יותר לשרת.

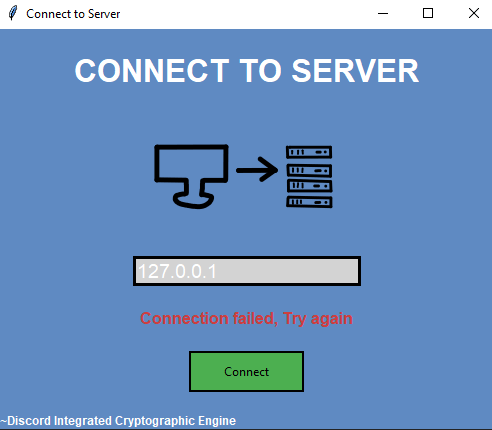
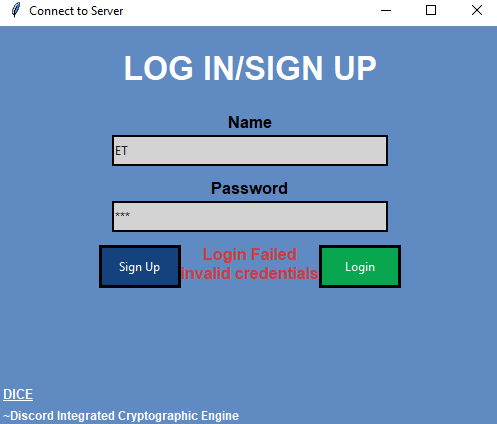
\*3 – אזור הקבצים האישיים: באזור הזה מוצגים כל הקבצים של הלקוח למשל פה מוצג קובץ טקסט עם השם "file name"

\*4 - כפתור המחיקה: הכפתור הזה ימחוק את הקובץ מדיסקורד, מסד הנתונים וגם מהתצוגה אצל הלקוח.

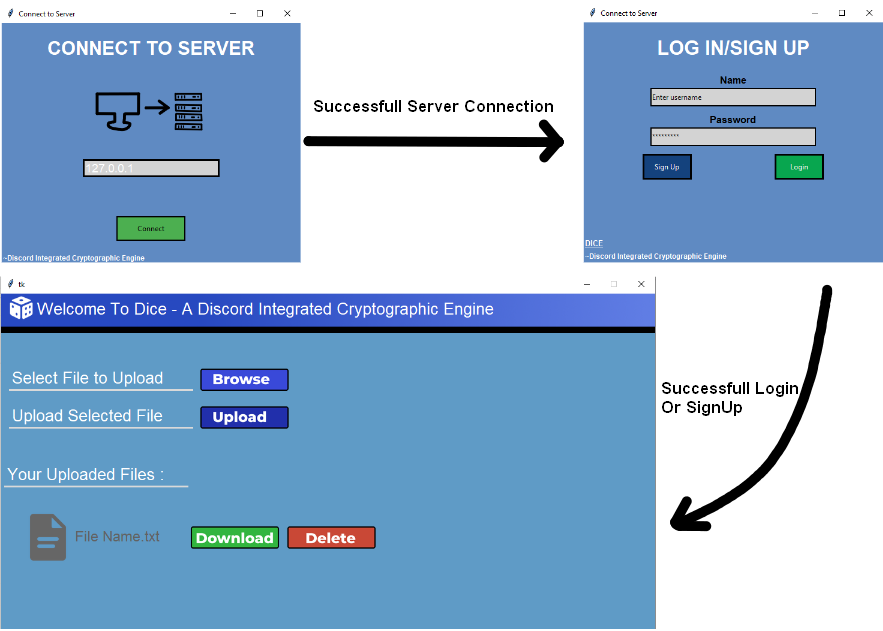
\*5 – כפתור ההורדה: הכתפתור ישלח בקשת הורדה של הקבוץ המבוקש לשרת. כתוצאה מכך הלקוח אמור לראות אצלו בתקיית הפרויקט את הקובץ החדש תחת השם "received\_file name"

**הודעות שגיאה במסכים השנונים:**במקרה שבו הקלט לא עבר בדיקה (למשל שם תפוס) או כל תקלה אחרת תופיע הודעת שגיאה ללקוח על המסך. לפניך דוגמה להודאות שגיאה במסכים:

**שגיאה בכניסה למערכת. שגיאה בהתחברות לשרת.**

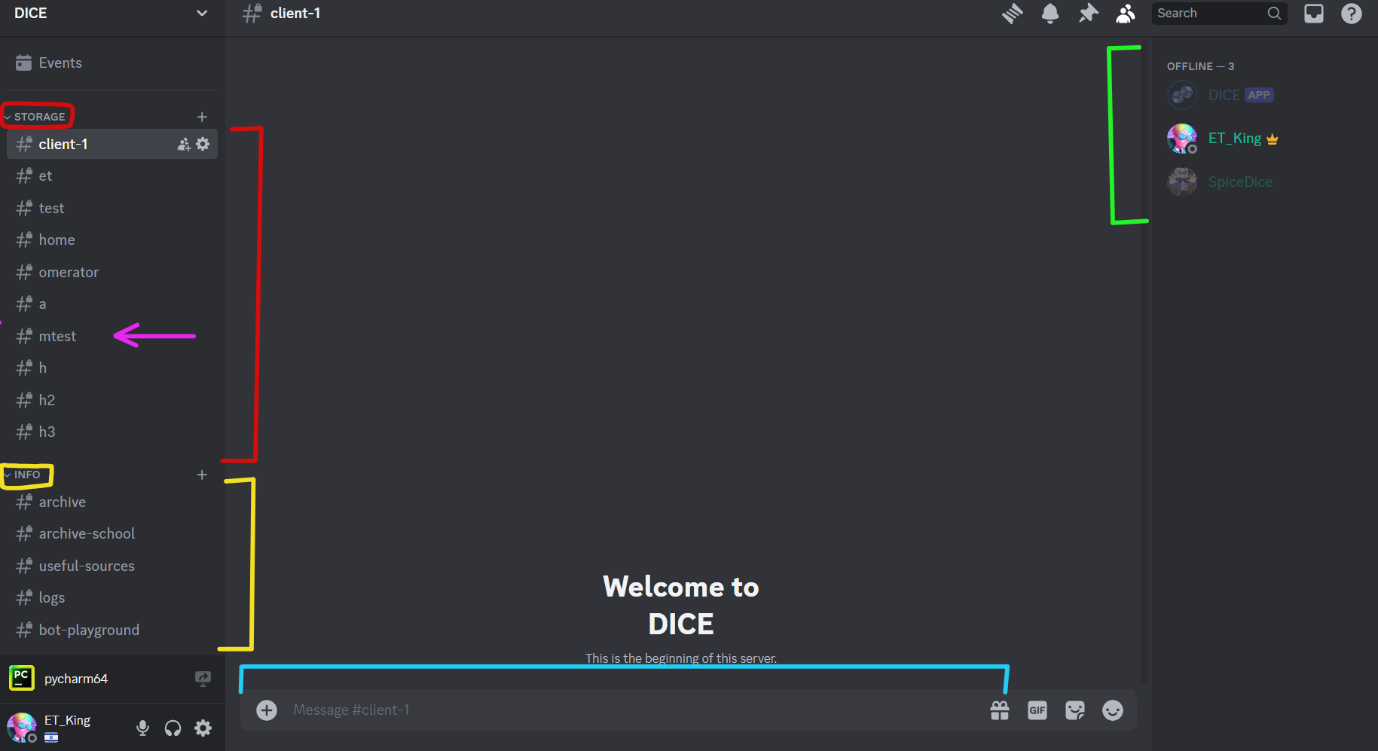


**תרשים זרימה של מסכי הפרויקט:**



**תיאור מבני הנתונים:**

בפרויקט זה מסד הנתונים הוא פשוט ביותר וכל מה שצריך לדעת עליו ניתן למצוא **פה.**

בניגוד למסד הנתונים שהוא פשוט מאוד יש לנו את העבודה מול דיסקורד (שגם היא פשוטה אך למי שלא מכיר את הפלטפורמה יכולה להיות קצת מפחידה בהתחלה). בואו נעשה הסבר קצר על איך להתנהל עם התוכנה. לאחר שהורדתם את התוכנה והתחברתם אל השרת הנדרש, אתם תראו מסך דומה לזה:

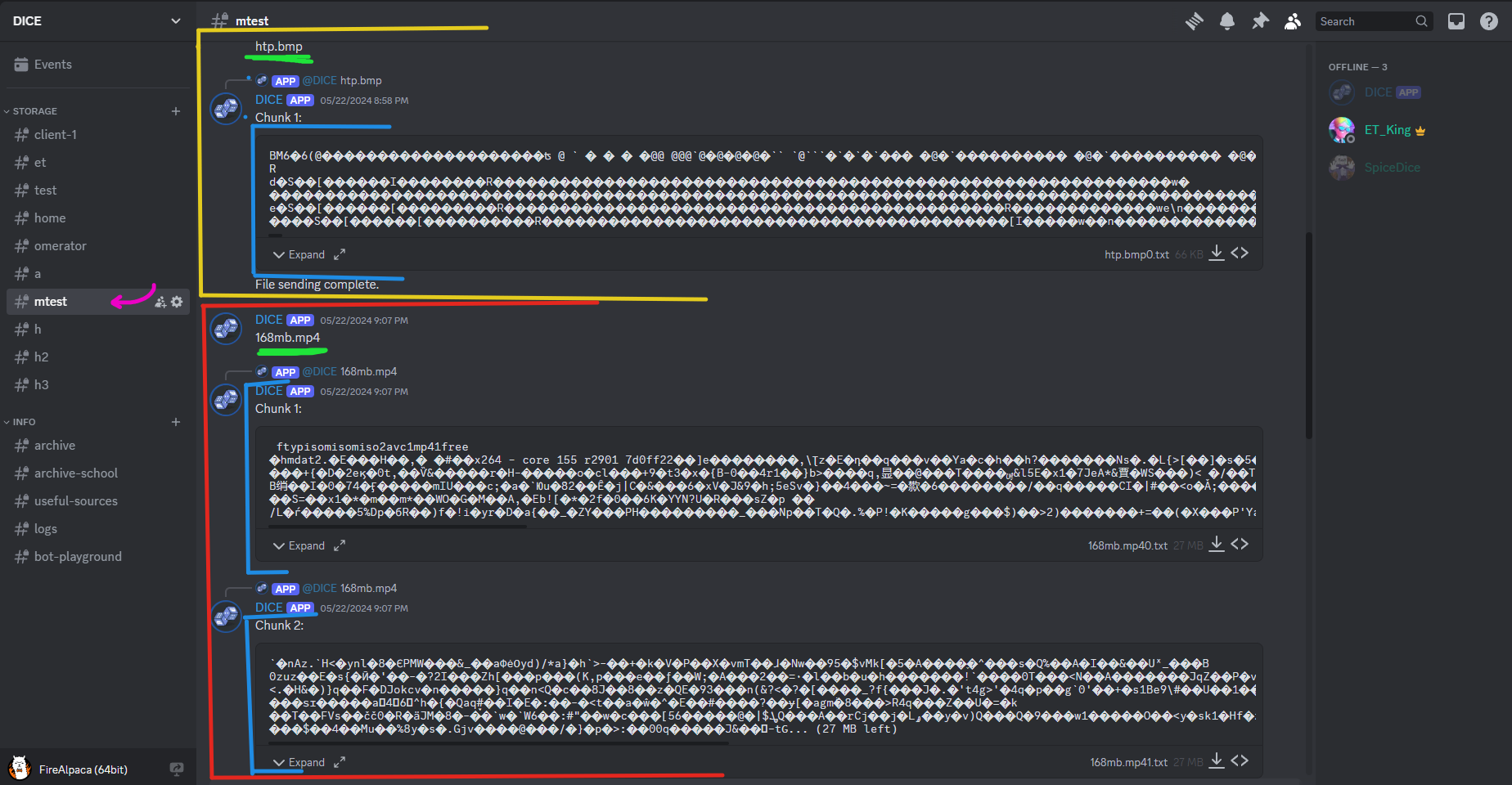
נעבור בקצרה על הדברים שמוצגים לנו כאן ונתעקב על הדברים החשובים.

בירוק – אנשים שמחוברים לשרת (וגם הבוט) – *לא מעניין*

בכחול – אופציה לכתוב הודעה – *לא מעניין*

בצהוב – אזור עם ערוצים ששימשו אותי במהלך הפרויקט בין אם זה לבדיקת הבוט, שמירת נתונים חשובים, העברת דברים מהבית ספר לבית ועוד. – *גם לא כזה מעניין*

באדום – אזור האחסון. זה כן אזור חשוב כל שורת טקסט פה מציגה ערוץ שאפשר להכנס אליו ע"י לחיצה עליו. כל ערוץ כזה מיצג לקוח ובפנים יש את הקבצים שלו. להמשך ההסבר נכנס לערוץ ­הוורוד -mtest ונראה מה יש לנו שם (\*כנראה שהערוצים לא ישארו זהים ותראו שם לקוחות אחרים)



כמו שאפשר לראות עכשיו אנחנו בערוץ של הלקוח mtest בתמונה אפשר לראות קובץ וחצי של הלקוח. בואו נפרק את זה לחלקים לאט לאט.

בצהוב – קובץ אחד בשם “htp.bmp”

באדום – חלק מקובץ אחר בשם "168mb.mp4" (רק חלק כי לא הכל נכנס בפריים)

בירוק – ההודה שמסמנת את הקובץ, זאת ההודעה ששומרים את ה message id שלה וכמו שניתן לראות זאת ההודעה שאליה מגיבים את חתיכות הקובץ

בכחול – החתיכות של הקבצים. כמו שאפשר לראות בקובץ הראשון (אזור צהוב) נדרש רק חלק אחד לקובץ כי הוא קטן יותר מ25mb ולכן הבוט מגיב להודעה פעם אחת עם חתיכה אחת בלבד. אך בניגוד לכך בקובץ השני (האדום) ניתן לראות ש25mb זה לא מספיק ולכן הוא מגיב ביותר מהודעה אחת, יותר מחתיכה אחת.

**סקירת חולשות ואיומים:**

הפרויקט כולל **Login**, **TCP**, **Database** **והצפנות**. בכל אלו יכולים להיות חולשות שניתן למנוע. כרגע בפרויקט אני נמנע מחלק מהדברים אך אני מתכנן להמשיך להוסיף הגנות מפני הדברים שאיניני מגן עלהים כרגע כמו למשל הגנה מפני sql injection ע"י בדיקת הקלט שלי, כל פעם שמועבר קלט של המשתמש למסד הנתונים הוא יעבור בדיקה כדי לוודא שאין שם נסיון לשנות את הפעולה המקורית. בתהליך ה**login** אני מוודא שהנתונים עומדים בקריטריונים הנדרשים עבור ההרשמה. בתהליך ההתחברות אני מוודא שאין כפילויות בשם כדי למנוע מצב שבו לשתי לקוחות יש את אותם הפרטים. **בפרוטוקול התקשורת** שלי, כל הפרוטוקול בנוי כעוד שכבה לTCP ולכן הבטיחות שאנחנו מקבלים מפרוטוקול זה מתווספת להצפנות שלנו.**ההצפנות**, בפרויקט יש כמה הצפנות. הראשונה היא הצפנת התקשורת, הצפנה זו נמצאת בפרויקט והורחב עליה בתחילת פרק זה. בנוסף התוכנית המקורית הייתה להצפין את הקבצים שמועלים לדיסקורד ע"י שימוש במפתח קבוע שנמצא אצל השרת, בזמן הכתיבה של הספר עדיין לא הגעתי לממש זאת, אך אני חדור במטרתי לסיים את הפרויקט כמו שחזיתי אותו ואני מניח שעד שיגיע הזמן להגיש את הפרויקט עצמו הצפנת הקבצים תהיה מוכנה.

# מימוש הפרויקט

**פירוט מימוש המערכת (חלק א')**

**מודלים/ מחלקות מיובאות**

מודלים הקשורים לעיצוב גרפי וממשק משתמש:

* import **tkinter** as tk – משמש בשביל יצירת ממשקי המשתמש
* from tkinter import **filedialog –** משמש עבור עבודה עם קבצים דרך ui
* from tkinter import **PhotoImage –** משמש בשביל עבודה עם תמונות וtk

מודלים הקשורים לדיסקורד

* import **discord -** הדרך העיקרית לתקשר עם דיסקורד (ניתן למצוא הסבר בנספחים)
* from discord import **Activity**, **ActivityType –** דרך לכתוב דברים על הבוט, לא משפיע על הפרויקט ישירות.

מודלים שקשורים למסד הנתונים

* import **sqlite3 –** משמש לתקשורת עם מסד הנתונים דרך Python.

מודלים בצד השרת והלקוח

* import **asyncio** – משמש עבור תכנות אסינכרוני.
* import **json** – משמש לעבודה עם פורמט JSON.
* import **base64** – משמש לקידוד ופענוח ב-Base64.
* import **socket** – משמש לתקשורת רשת באמצעות סוקט.
* import **threading** – משמש לניהול תהליכונים (threads).
* import **os** – משמש לעבודה עם מערכת הקבצים והפעלה של פקודות מערכת.
* import **time** – משמש לניהול זמן והשגת נתונים הקשורים לזמן.

מודלים שקשורים בהצפנה

* from cryptography.hazmat.backends import **default\_backend** – משמש כברירת מחדל לגיבוי בטכנולוגיית הצפנה.
* from cryptography.hazmat.primitives import **serialization**, **hashes** – משמש לסדרת כלים לצפנה סימטרית וצפנה בכללי.
* from cryptography.hazmat.primitives.asymmetric import **rsa**, **padding** – משמש לסדרת כלים לצפנה אסימטרית, כולל RSA ופדינג.
* from Crypto.Util.Padding import pad, **unpad** – משמש לפדינג ואנפדינג בצפנה סימטרית.
* from Crypto.Random import **get\_random\_bytes** – משמש ליצירת מספרים אקראיים לצורך צפנה סימטרית.

\*הרוב הם דפנדנסיז שלא הייתה לי ברירה להוריד.

מודלים נוספים

* import **io** – משמש לקריאה וכתיבה בצורה בסיסית מאוד על אובייקטים בפייתון, דוגמת קבצים.
* import **requests** – משמש לשליפת תוכן מאתרים דרך HTTP/HTTPS.
* import **os** – משמש לפעולות עם מערכת הקבצים והסביבה של המערכת ההפעלה.

**מחלקות שאני פיתחתי**

**שם המחלקה:** Database

**תפקיד המחלקה:**  מחלקה זו מיועדת לניהול מסד הנתונים של המשתמשים והקבצים. היא מטפלת בהתחברות למאגר הנתונים, יצירת חשבונות חדשים, וניהול קבצים בערוצים שונים. המחלקה גם מספקת פעולות לשליפה ועדכון של נתונים ממאגר המידע.

**תכונות המחלקה:**

* db\_name – שם מסד הנתונים (משמש לאחסון שם הקובץ של מסד הנתונים).
* conn – חיבור למסד הנתונים (משמש לניהול חיבור פעיל למסד הנתונים).
* cursor – סמן למסד הנתונים (משמש לביצוע פעולות שאילתא על מסד הנתונים).

**פעולות המחלקה:**

* **\_\_init\_\_(self, db\_name)**

כניסה: שם מסד הנתונים. יציאה: אתחול חיבור וסמן למסד הנתונים.

* **attempt\_login(self, username, password)**

כניסה: שם משתמש וסיסמה. יציאה: רשומת משתמש אם קיימת התאמה.

* **is\_username\_availability(self, username)**

כניסה: שם משתמש. יציאה: True אם פנוי, False אם לא.

* **create\_new\_account(self, username, password, channel\_id)**

כניסה: שם משתמש, סיסמה, מזהה ערוץ. יציאה: יצירת חשבון וערוץ חדש במסד הנתונים.

* **create\_new\_file\_table(self, channel\_id)**

כניסה: מזהה ערוץ. יציאה: יצירת טבלת קבצים חדשה.

* **new\_file\_in\_channel(self, filename, message\_id, channel\_id)**

כניסה: שם קובץ, מזהה הודעה, מזהה ערוץ. יציאה: הוספת קובץ חדש לערוץ במסד

* **get\_files\_from\_id(self, channel\_id)**

כניסה: מזהה ערוץ. יציאה: רשימת קבצים מהערוץ.

* **get\_message\_id\_by\_name(self, filename, channel\_id)**

כניסה: שם קובץ, מזהה ערוץ. יציאה: מזהה הודעה עבור הקובץ.

* **delete\_file\_in\_table(self, filename, channel\_id)**

כניסה: שם קובץ, מזהה ערוץ. יציאה: מחיקת הקובץ מהערוץ במסד הנתונים.

**שם המחלקה:** DiscordBot

**תפקיד המחלקה:** מחלקה זו מיועדת לניהול בוט של דיסקורד המשתמש ב-API של דיסקורד לביצוע פעולות שונות. המחלקה מאפשרת שליחה, מחיקה והרכבה של קבצים בערוצים, יצירת ערוצים חדשים, וביצוע פעולות נוספות על פי פקודות המשתמשים. בנוסף, המחלקה מאפשרת דיבוג דרך הצ'אט.

**תכונות המחלקה:**

* token – מחרוזת האימות של הבוט (משמשת לאימות והתחברות לשירות דיסקורד).
* intents – אובייקט הכוונות של דיסקורד (משמש להגדרת ההרשאות והכוונות של הבוט).
* bot – אובייקט הלקוח של דיסקורד (משמש להפעלת הבוט והתקשורת עם שרתי דיסקורד).
* file\_manager – אובייקט של מנהל הקבצים (משמש לניהול פעולות הקשורות לקבצים, הסבר בהמשך).

**פעולות המחלקה:**

* **\_\_init\_\_(self, token) –**

כניסה: מחרוזת אימות של הבוט. יציאה: אתחול הכוונות, יצירת אובייקט לקוח של דיסקורד ומנהל קבצים.

* **get\_available\_file\_name(self, file\_name, channel\_id) –**

כניסה: שם קובץ, מזהה ערוץ. יציאה: שם קובץ פנוי שלא קיים בערוץ.

* **create\_new\_storage\_area(self, channel\_name) –**

כניסה: שם ערוץ. יציאה: מזהה הערוץ החדש שנוצר.

* **get\_message\_id\_by\_content(self, channel\_name, message\_content) –**

כניסה: שם ערוץ, תוכן הודעה. יציאה: מזהה ההודעה עבור התוכן שנמצא בערוץ.

* **on\_ready(self) –**

כניסה: None. יציאה: מדפיס הודעה ומעדכן את הסטטוס של הבוט.

* **send\_file\_in\_chat(self, file\_name, file\_content, channel\_id) –**

כניסה: שם קובץ, תוכן קובץ, מזהה ערוץ. יציאה: שולח קובץ בערוץ ומחזיר את שם הקובץ והמזהה.

* **assemble\_file\_from\_chat(self, message\_id, channel\_id) –**

כניסה: מזהה הודעה, מזהה ערוץ. יציאה: מרכיב קובץ מהודעות בערוץ ומחזיר את הנתונים.

* **delete\_file\_from\_chat(self, message\_id, channel\_id) –**

כניסה: מזהה הודעה, מזהה ערוץ. יציאה: מוחק את הקובץ והודעות הקשורות בערוץ.

* **on\_message(self, message) –**

כניסה: אובייקט הודעה של דיסקורד. יציאה: מטפל בהודעות שנשלחו לבוט ומבצע פעולות בהתאם לפקודות.

* **run\_discord\_bot(self) –**

כניסה: None. יציאה: מפעיל את הבוט ומאזין לאירועים.

**שם המחלקה:** FileManager

**תפקיד המחלקה:** מנהל קבצים במערכת, מטפל בפעולות קשורות לטיפול, הפצה ואספקת קבצים. המחלקה מספקת פונקציות לחלוקת קבצים לחלקים ולאיסוף חלקי הקבצים לקובץ אחד.

**תכונות המחלקה:**

CHUNK – גודל החתיכה לחלוקת הקבצים, בבתים.

**פעולות המחלקה:**

* **split\_file\_data(self, file\_data: bytes)**

כניסה: נתוני קובץ בתוספת bytes. יציאה: רשימת חלוקות של נתוני הקובץ.

* **assemble\_file(self, sorted\_reply\_messages)**
* כניסה: רשימת הודעות ממוינות. יציאה: הודעת קובץ שנאספה מההודעות הנתונות.

**שם המחלקה:** AsymmetricEncryptionProtocol

**תפקיד המחלקה:** מחלקת AsymmetricEncryptionProtocol מיועדת להפעיל פרוטוקולים של הצפנה אסימטרית, בעיקר בהשתמשות באלגוריתמים של RSA היא מאפשרת ליצור מפתחות פרטיים וציבוריים, להצפין ולפענח נתונים, ולהצפין מפתחות סימטריים באמצעות מפתחות ציבוריים.

**תכונות המחלקה:**

* private\_key (מפתח פרטי): מכיל את המפתח הפרטי של השרת לצורך פענוח נתונים.
* public\_key (מפתח ציבורי): מכיל את המפתח הציבורי של השרת לצורך הצפנת נתונים

**פעולות המחלקה:**

* **get\_public\_key\_as\_str:**

כניסה: public\_key - המפתח הציבורי להמיר לתבנית PEM

יציאה: מחרוזת שמייצגת את המפתח הציבורי בתבנית PEM

* **load\_public\_key\_from\_:**

כניסה: pem\_str - המחרוזת שמכילה את המפתח הציבורי בתבנית PEM

יציאה: אובייקט המייצג את המפתח הציבורי המטען מהמחרוזת

* **create\_server\_keys:**

כניסה: אין

יציאה: אין

* **get\_public\_key:**

כניסה: אין

יציאה: המפתח הציבורי של השרת בתבנית PEM

* **decrypt\_data :**

כניסה: encrypted\_data - הנתונים המוצפנים לפענוח

יציאה: הנתונים המפוענחים

* **encrypt\_data:**

כניסה: data - הנתונים להצפנה

יציאה: הנתונים המוצפנים

* **encrypt\_symmetric\_key:**

כניסה: symmetric\_key - המפתח הסימטרי להצפנה, server\_public\_key - המפתח הציבורי של השרת להצפנה

יציאה: המפתח הסימטרי המוצפן

**שם המחלקה:** SymmetricEncryptionProtocol

**תפקיד המחלקה:** מחלקת SymmetricEncryptionProtocol מיועדת לביצוע פעולות הצפנה ופענוח באמצעות אלגוריתם הצפנה סימטרי כמו AES. היא מאפשרת להצפין ולפענח נתונים על ידי יצירת מפתחות וכן שימוש ב-IV (Initialization Vector) לצורך יצירת סיסמה מאובטחת.

**תכונות המחלקה:**

אין, המחלקה נותנת פעולות סטטיות.

**פעולות המחלקה:**

* **encrypt\_data:**

כניסה: key - המפתח להצפנה, data - הנתונים להצפנה

יציאה: הנתונים המוצפנים יחד עם ה-IV שנוצר

* **decrypt\_data:**

כניסה: key - המפתח לפענוח, encrypted\_data - הנתונים המוצפנים

יציאה: הנתונים המפוענחים

* **encrypt\_packet:**

כניסה: key - המפתח להצפנה, packet - החבילה (מחרוזת) להצפנה

יציאה: החבילה המוצפנת באמצעות הצפנה סימטרית

* **decrypt\_packet:**

כניסה: key - המפתח לפענוח, packet - החבילה (מחרוזת) המוצפנת

יציאה: הנתונים המפוענחים מהחבילה

\*המחלקות הבאות הן מחלקות שאני עדיין רוצה להוסיף ולשנות בהם דברים ולכן יתכן שהן לא יהיו זהות למה שמוצג כאן בסופו של דבר.

**שם המחלקה:**  Client

**תפקיד המחלקה:** מחלקת Client משמשת לניהול תקשורת עם שרת מרוחק. היא מטפלת בחיבור לשרת, שליחת ובקשת קבצים, ביצוע פעולות ההתחברות והרישום, והצפנה ופענוח של נתונים באמצעות פרוטוקולים אסימטריים וסימטריים. המחלקה גם שימוש בפרוטוקולי ההצפנה.

**תכונות המחלקה:**

* aes: מטפל בהצפנה ופענוח סימטריים.
* symmetric\_key: המפתח להצפנה סימטרית.
* asymmetric\_protocol\_instance: מטפל בהצפנה אסימטרית.
* host: כתובת השרת.
* port: פורט השרת.
* user\_data: מידע על המשתמש המחובר.
* users\_files: אוסף קבצי המשתמשים.
* reload\_files\_flag: מציין אם יש לטעון קבצים מחדש.
* packet\_handlers: רשימה של פונקציות לטיפול בחבילות שונות מהשרת.
* **פעולות המחלקה:**connect\_to\_server:

כניסה: אין

יציאה: בוליאני המציין אם החיבור לשרת הצליח

* **receive\_data:**

כניסה: אין

יציאה: אין

* **get\_file\_from\_server:**

כניסה: data\_length - אורך הנתונים

יציאה: אין

* **send\_file\_to\_server:**

כניסה: file\_path - נתיב הקובץ לשליחה, channel\_id - מזהה הערוץ

יציאה: אין

* **request\_signup:**

כניסה: username - שם משתמש, password - סיסמה

יציאה: בוליאני המציין אם הרישום הצליח

* **request\_file\_deletion:**

כניסה: file\_name - שם הקובץ למחיקה, channel\_id - מזהה הערוץ

יציאה: אין

* **request\_download\_file:**

כניסה: file\_name - שם הקובץ להורדה, channel\_id - מזהה הערוץ

יציאה: אין

* **request\_login:**

כניסה: username - שם משתמש, password - סיסמה

יציאה: אין

* **request\_user\_files:**

כניסה: אין

יציאה: אין-

* **handle\_files\_for\_initiation:**

כניסה: data\_length - אורך הנתונים

יציאה: אין

* **handle\_login\_data:**

כניסה: data\_length - אורך הנתונים

יציאה: אין

* **handle\_server\_key:**

כניסה: data\_length - אורך הנתונים

יציאה: אין

* **zero\_fill\_length:**

כניסה: input\_string - מחרוזת קלט, width - רוחב (ברירת מחדל 4)

יציאה: מחרוזת באורך קבוע עם אפסים למילוי

**שם המחלקה** :Server

**תפקיד המחלקה:** מחלקת Server משמשת לניהול תקשורת עם לקוחות מרוחקים. היא מאזינה לחיבורים נכנסים, מטפלת בבקשות מהלקוחות, ומנהלת העברת קבצים. היא מבצעת תקשורת באמצעות פרוטוקולים אסימטריים וסימטריים. בנוסף, המחלקה מתקשרת עם בוט דיסקורד לניהול קבצים בערוצי דיסקורד.

**תכונות המחלקה:**

* asymmetric\_protocol\_instance: אינסטנס של פרוטוקול ההצפנה האסימטרי.
* host: כתובת השרת.
* port: פורט השרת.
* clients: מילון לשמירת חיבורי הלקוחות.
* clients\_aes: רשימה של מפתחות הצפנה סימטריים של הלקוחות.
* client\_threads: רשימה של חוטי לקוחות.
* packet\_handlers: רשימה של פונקציות לטיפול בחבילות שונות מהלקוחות.
* bot\_instance: אינסטנס של בוט דיסקורד.

**פעולות המחלקה:**

* **start:**

כניסה: אין

יציאה: אין

* **run\_server\_for\_client:**

כניסה: client\_id

יציאה: אין

* **receive\_data:**

כניסה: client\_id

יציאה: bool

* **receive\_file:**

כניסה: data\_length, client\_id

יציאה: list

* **send\_files\_to\_discord:**

כניסה: file\_name, file\_content, channel\_id, client\_id

יציאה: אין

* **handle\_file\_and\_send\_to\_discord:**

כניסה: data\_length, client\_id

יציאה: אין

* **handle\_sign\_up\_request:**

כניסה: data\_length, client\_id

יציאה: אין

* **handle\_file\_request:**

כניסה: data\_length, client\_id

יציאה: אין

* **handle\_deletion\_request:**

כניסה: data\_length, client\_id

יציאה: אין

* **handle\_login\_request:**

כניסה: data\_length, client\_id

יציאה: אין

* **handle\_files\_for\_initiation:**

כניסה: data\_length, client\_id

יציאה: אין

* **handle\_symmetric\_key:**

כניסה: data\_length, client\_id

יציאה: אין

* **recvall:**

כניסה: client\_id, buffsize

יציאה: bytes

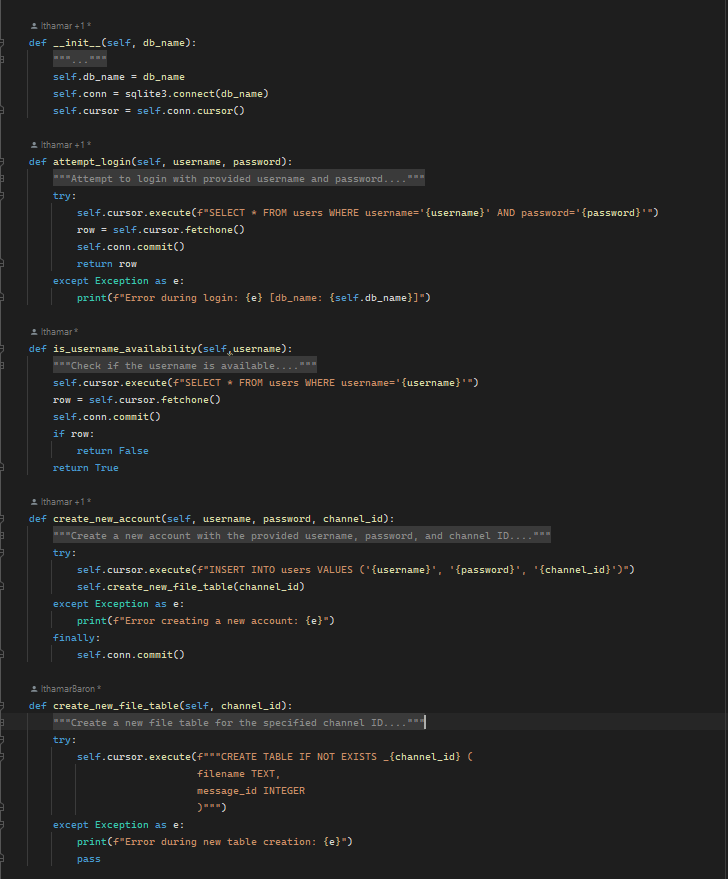
* **zero\_fill\_length:**

כניסה: input\_string, width (default 4)

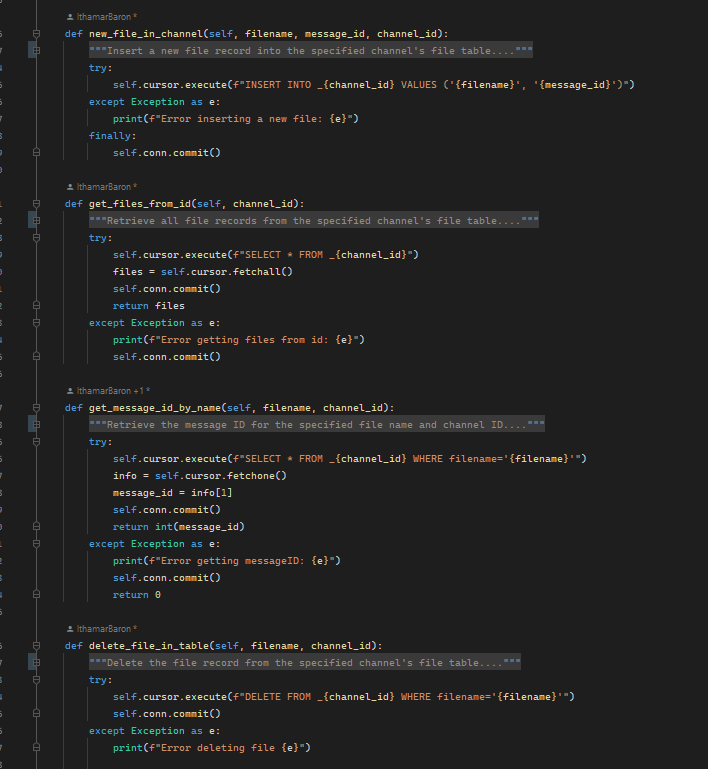
יציאה: str

**פירוט מימוש המערכת (חלק ב)**

\*בחלק זה מתבקשים להציג קטעי קוד. למען אסטטיות אציג את קטעי הקוד בתמונות, **אך ניתן למצוא את כל הקוד שמוצג פה גם כטקסט בנספחים.**

בואו נתחיל בלהציג את הקוד מהפשוט למסובך. תחילה נתחיל מבסד הנתונים. הנה הפעולות שאחראיות על תהליך ההתחברות והרשמה, ניתן למצוא הסבר על כל אחת מהן למעלה בחלק א' 

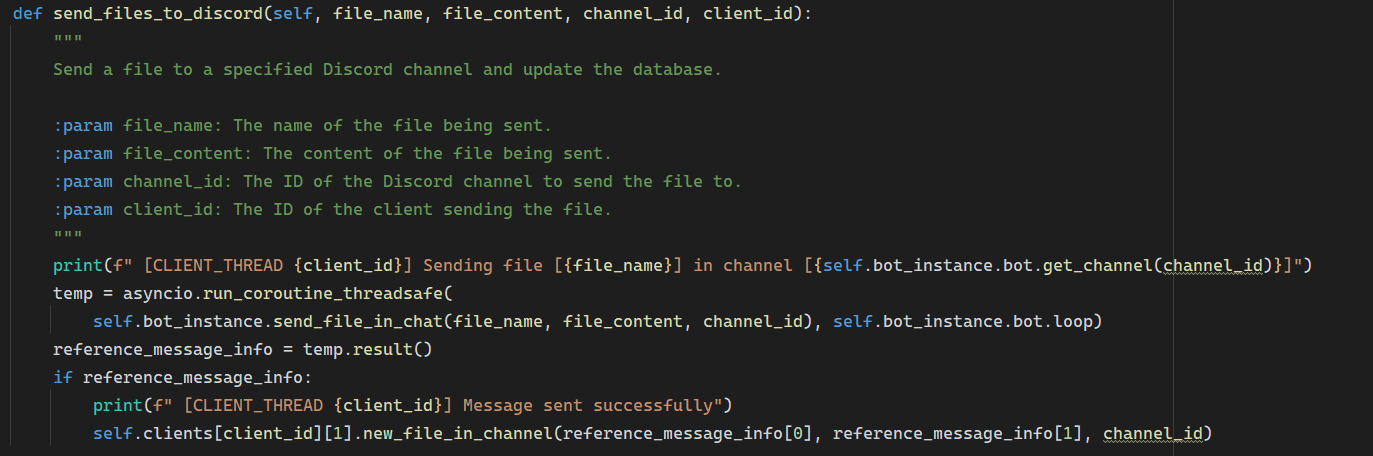
הנה הפעולות שאחריות על טיפול במידע על הקבצים של כל לקוח, גם עליהן ניתן למצוא הסברים בחלק א'



\*בתמונות קיצרתי את ההערות של כל פעולה מכיוון שלא יכלתי להכניס את הכל בתמונה אך ניתן לקרוא אותם בנספחים.

**תהליך שמירת הקבצים**

בחלק זה אראה חלקי קוד שמתאימים לתרשים הזרימה "תרשים זרימה – שליחת קובץ" מפרק המבוא.

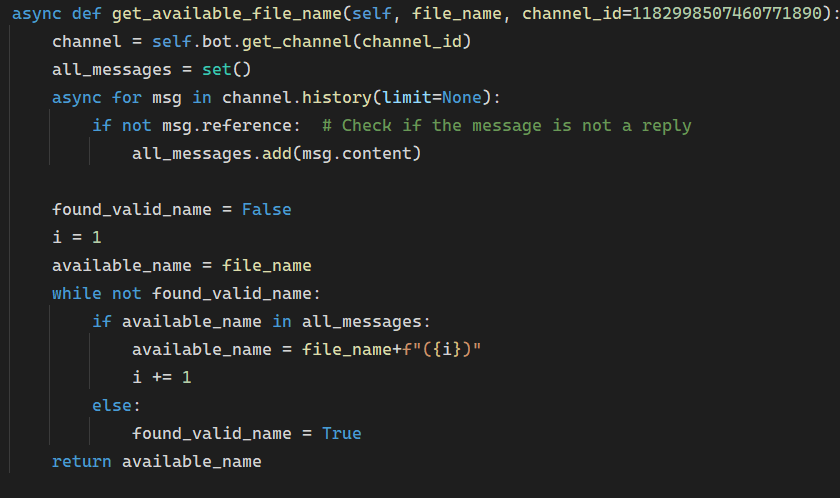
המצב ההתחלתי שלנו בתרשים הוא שלשרת כבר יש את קובץ הלקוח מוצפן לכך נמשיך משם. 

השרת משתמש בפעולה זו שבתוכה היא קוראת לפועלה send\_file\_in\_chat:

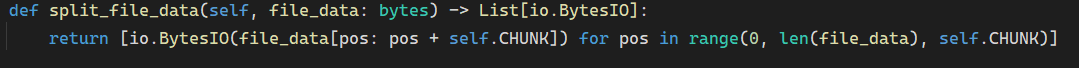
A computer screen shot of a program

Description automatically generated

הפעולה עושה כמה דברים. ראשית היא בודקת את זמינות השם עם הפעולה get\_available\_file\_name הפעולה נראת כך והיא תחזיר את השם הזמין עבור הקובץ



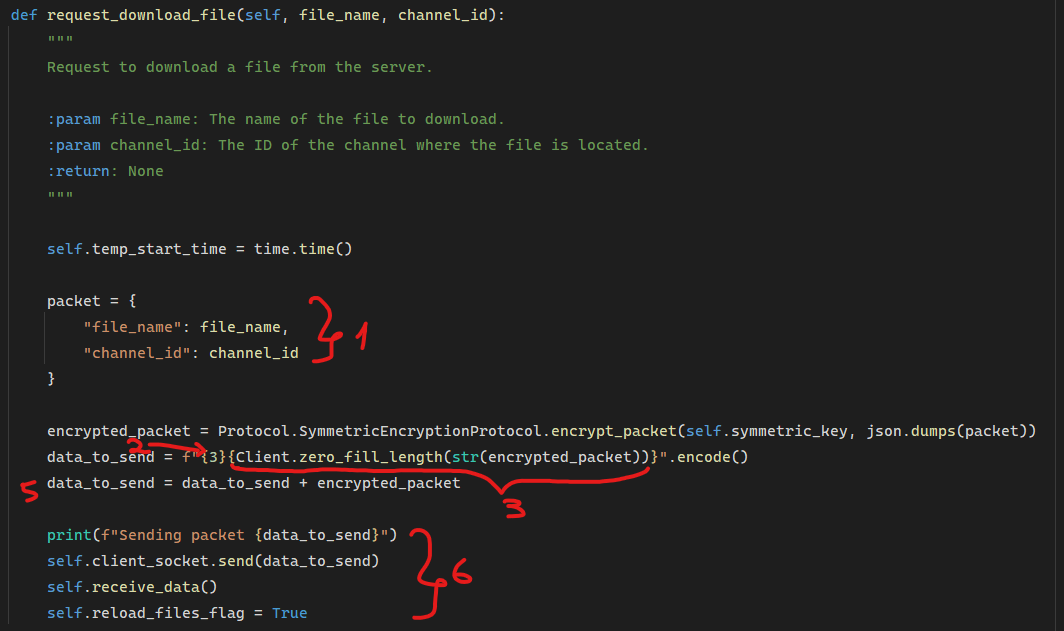
לאחר שקיבלנו את השם הזמין לקובץ אנחנו חוזרים חזרה ואז הפעולה הזאת קוראת לsplit\_file\_data אשר מפצלת את הקבצים והיא נראת ככה:



הפעולה send\_file\_in\_chat משתמשת בחתיכות הקובץ ועוברת עליהם ואז שולחת אותם בערוץ הדיסקורד המתאים. לבסוף היא מחזירה את השם הזמין כדי לעדכן את מסד הנתונים והלקוח ביחד עם מזהה ההודעה שאליה הגבנו. (חזרנו חזרה לפעולה הראשונה עכשיו) אם קיבלנו באמת דברים חזרה אז אנחנו מעדכים את מסד הנתונים עם השם של הקובץ מזהה ההודעה.

**פרוטוקול הפרויקט**

כמו שכבר ראינו הפרוטוקול משתמש במזהה פקטה יחודי עבור כל פקטה. בואו נראה דוגמה לאיך מועברת פקטה (נדקים על פקטת ההתחברות)



1 – המידע שאנחנו רוצים להעביר

2 – הגדרת מזהה הפקטה

3 – הוספת אורך הפקטה באופן שיתפוס ארבעה מספרים

4 – התבלבלתי בציור ;)

5 - חיבור המזהה האורך והמידע המוצפן

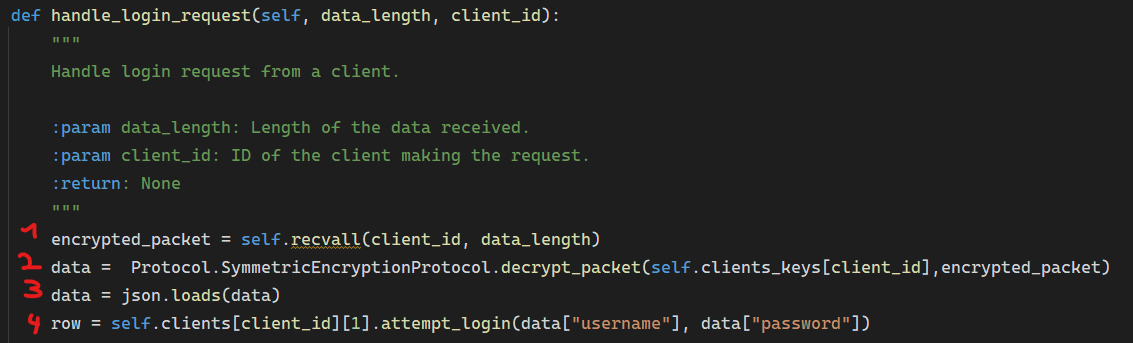
6 – שליחת הפקטה במבנה המצופה

עכשיו בואו נראה מה קורה בשלב קבלת המידע בצד השרת.

A screen shot of a computer code

Description automatically generated

1. קריאת מזהה הפקטה
2. קריאת אורך המידע
3. קריאה לפעולת התפעול הנכונה בהתבסס על שמזהה הפקטה עם הפרמטרים 'אורך המידע' להמשך טיפול ומזהה הלקוח למטרות זיהוי



עכשיו קראנו לפעולה המתאימה לפי מזהה הפקטה. בתוכה אנחנו:

1. קוראים את המידע המוצפן מהסוקט
2. מפענחים את הפקטה
3. מוצאים את אוביקט המילון מפורמט הגייסון
4. משתמשים במידע בפקטה

**מסמך בדיקות מלא:**

**שם הבדיקה:** הרשמה של מתשמש חדש.

**מטרת הבדיקה:**  ליצור משתמש חדש במערכת באופן שיאפשר התחברות חוזרת אליו.

**ביצוע הבדיקה:** יצירת משתמש בכמה דרכים שגויות ואז בדרך אחת נכונה ולראות שכל שאר הדרכים לא מתקבלות (למשל שם המשתמש תפוס) .

**תוצאות הבדיקה:** המערכת גילתה שהשם תפוס ולא אישרה את הכניסה אך כן אישרה משתמש שהתחבר בדרך הרצויה.

**שם הבדיקה:** כניסה למערכת.

**מטרת הבדיקה:** להכנס למשתמש רשום במערכת.

**ביצוע הבדיקה:** לנסות להכנס עם פרטים שגוים/ פרטים לא קיימים ואז לנסות עם פרטים נכונים.

**תוצאות הבדיקה:** המערכת לא תיתן להכנס אם פרטי הלקוח אינם נכונים ורשומים במערכת.

**שם הבדיקה:** חיבור כמה לקוחות במקביל.

**מטרת הבדיקה:** לראות שהשרת יכול להתמודד ולנהל תקשורת בין 2 לקוחות בו זמנית.

**ביצוע הבדיקה:** הרמת שתי או יותר לקוחות וחיבורם במקבים לשרת, ביצוע תקשורת.

**תוצאות הבדיקה:** השרת יקבל את שתי הלקוחות ויגיב להם בצורה מסונכרנת.

**שם הבדיקה:** שליחת קובץ גדול מ25mb

**מטרת הבדיקה:** לבדוק שניתן לחמוק מההגבלה שדיסקורד מציבים על גודל הקובץ.

**ביצוע הבדיקה:** לאחר התחברות לשרת ולמערכת. יש לבחור קובץ הגדול מ25mb וקטן מ1gb (הגבלה שאני שמתי) ולהעלות אותו לשרת.

**תוצאות הבדיקה:** הקובץ ישלך בדיסקורד בחלקים של 25mb.

**שם הבדיקה:** הורדת קובץ הגדול מ25mb.

**מטרת הבדיקה:** לבדוק שאנחנו מצליחים להשיג את כל החתיכות ולחברן בלי לאבד מידע בדרף.

**ביצוע הבדיקה:** במידה והועלה בעבר קובץ הגדול מ25mb יש לנסות להוריד אותו. אם לא, יש להעלות קובץ ואז לנסות להוריד אותו

**תוצאות הבדיקה:** הקובץ יופיע בתקיית הפרויקט ויפתח כמצופה ממנו.

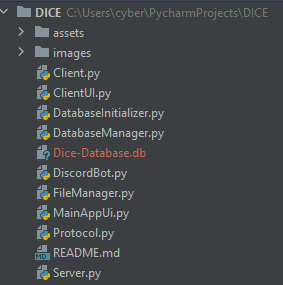
**שם הבדיקה:** מחיקת קובץ

**מטרת הבדיקה:** לבדוק האם ניתן למחוק קובץ מDICE

**ביצוע הבדיקה:** אם הועלה קובץ בעבר יש ללחוץ על כפתור delete (האדום). אם לא יש להעלות קובץ ואז לעשות זאת. לאחר מכן יש לחכות שיתעדכן המסך

**תוצאות הבדיקה:** הקובץ וההודעות שקשורות לקובץ ימחקו משרת הדיסקורד, הזיכרון לקובץ הזה ימחק ממסד הנתונים והמשתמש יעודכן ע"י מחיקת סימן הקובץ הזה בממשק המשתמש.

# מדריך למשתמש

**עץ הקבצים**

יש לציין שהקובץ שמופיע באדום כנראה לא יופיע

אלכם. ובשביל ליצור אותו יש להריץ את הקובץ "DatabaseInitializer.py" קובץ זה יצור מסמך שישמש כמסד נתונים **והוא הכרחי** להרצת הפרויקט.

\*לפה גם יגיעו הקבצים שמורידים מהמערכת.

**הכלים הנדרשים לבדיקת הפרויקט:**

Python – להרצת כל קבצי הפרויקט.

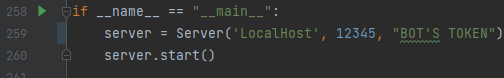
PIP – בשביל להוריד את כל החבילות הנדרשות כמו Discord ו Cryptodome

Discord – אינו **נחוץ** להרצת הפרויקט אך נחוץ על מנת לראות מה באמת קורה מאחורי הקלעים.

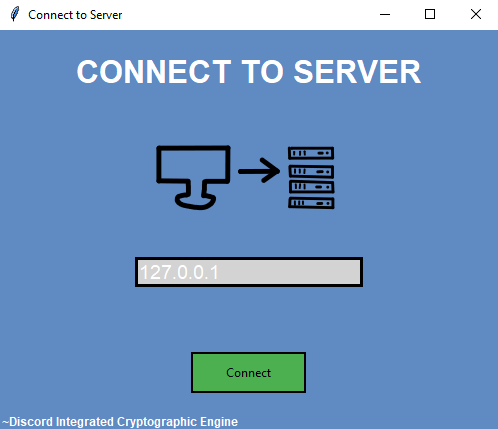
ניתן להוריד דיסקורד [בקישור הזה](https://discord.com/download). לאחר ההורדה יש ליצור משתמש ולהתחבר בשרת הדיסקורד של המערכת עי לחיצה על [הלינק הזה](https://discord.gg/jWYtW6An), ניתן למצוא הסבר על המסך שהגעתם עליו כעת ב"הסבר על מסכי המערכת"

**איך להריץ את הפרויקט**

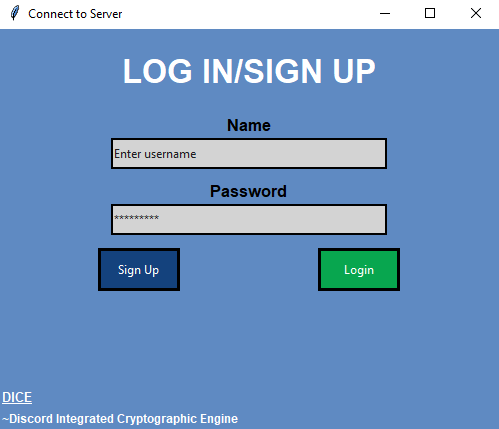
על מנת להריץ את הפרויקט יש להריץ את **Server** ואת **ClientUI**

**חשוב מאוד לציין** שבשרת נמצאת השורה הזו, בשורה זו מכניסים את פרטי האתחול של השרת

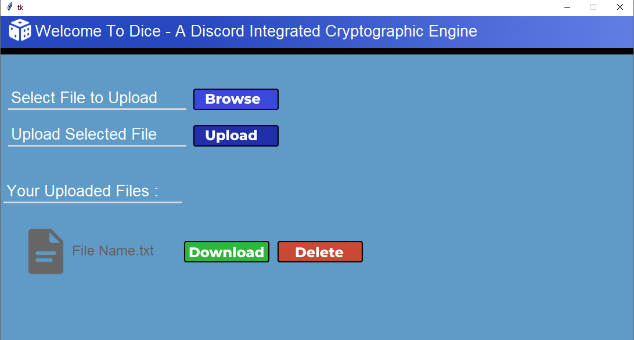
חשוב מאוד ש Bot's Token יהיה הטוקן הזה: REMOVED

בנוסף, חשוב מאוד שהטוקן יועתק אחד לאחד ו**רק** לפרויקט בתוך פאייצ'ארמ. לא ניתן לשים אותו בשום מקום עם גישה לאינטרנט בגלל המנגנון של דיסקורד שמונע הדלפות של טוקנס. למשל אם הקוד הכולל את הטוקן מועלה לגיט האב, דיסקורד בצורה אוטומטית ישנו את הטוקן ולא תהיה לי דרך לעדכן אתכם בכך.

לאחר שהרצתם את client ui ואת server אתם אמורים לראות את המסך הזה. כעת יש לשנות את כתובת האייפי במידת הצורך. לאחר שינוי כתובת האייפי יש ללחוץ על כפתור הconnect. במידה ומופיעה הודעת שגיאה בצג הגרפי, יש לוודא שהשרת אכן רץ כראוי או שהכתובת איפי לא נכונה.



במידה והכל עובד כראוי כעת תועברו למסך ההרשמה או התחברות אשר נראה כך:

במסך זה תדרשו ליצור משתמש מכיוון וזאת פעם ראשונה שאתםמשתמשים במערכת. בשביל לעשות זאת יש להכניס שם תחת הכותרת NAME וסיסמה (למשל pas123) תחת הכותרת PASSWORD. לאחר שהכנסת את הפרטים יש ללחוץ על כפתור ה signup המופיע בצד שמאל של המסך. במידה וההתחברות הייתה תקינה המערכת תחבר אתכם ישר. במידה ולא יש לעקוב אחרי ההוראות בהודעת השגיא אשר תוצג במסך המשתמש (למשל שם תפוס)

כעת תועברו למסך הראשי כאן ניתן להעלות קובץ ע"י לחיצה על browse בחירת קובץ ולחיצה על upload (יש סיכוי שהמסך שיופיע יהיה שונה מהמסך הזה בגלל בעיות שיש לי לאחרונה בקוד. המסך השני משמש אותה פעולה פשוט נראה פחות טוב, הוא יותר "debug-pannel" גם בו יש את כפתורי הbrowse ו upload אך יכול להית שיהיו להם שמות מעט שונים) לאחר העלאת הקובץ (שקטן מ1gb) ניתן לבצע בו פעולות כמו הורדה ומחיקה עי לחיצה על הכפתורים המתאימים במסך. בנוסף ניתן לראות את הקובץ שנשלח בדיסקורד בחלקים (אם גדול מ25MB)

# רפלקציה

**התמודדות עם קשיים במהלך הפרויקט**

במהלך ספר זה דיברתי רבות על האתגרים, הקשיים ודרכי הפתרון. הסברתי על כל אחד מהקשיים לעומק וגם על דרך הפתרון. אך בכל זאת lets recap.

**הגבלת דיסקורד** - דיסקורד מגבילים את העלעת הקבצים ל25mb . הדרך שבה אנחנו פותרים את זה היא פיצול הקבצים לפני העלעתם לחתיכות שכל אחת מהן היא 25mb וכך אנו נמנעים מלעבור על הגבלה זו באופן יצירתי.

**התלות בדיסקורד** – בכל רגע נתון דיסקורד יכולים לשנות משהו, כמו שראינו הם שינו אך למזלנו זה לא השפיע עלינו (שינוי אופן הלינק שדרכו מורידם דברים מהשרתים של דיסקורד)

**בעית הזמן** – על בעיה זו לא דיברתי, אך כן חשוב לי לדבר עליה. כאשר התחלתי את הפרויקט הייתי חדור מוטיבציה ועבדי עליו בתשוקה רבה, אך לאט לאט התחילו לבוא המבחנים, המיונים והעומסים מכל עבר... ולכן לא יכלתי להשקיע את הזמן שרציתי להשקיע בפרויקט. כתוצאה מכך הרבה דברים התעקבו וזאת אחת הסיבות שכרגע (בזמן שאני כותב את זה) לא כל הפיצ'רים מוכנים. אך אני מקווה מאוד שאני אצליח לסיים הכל בזמן ולהגיש פרויקט ברמת גימור גבוהה, ולא רק שיעמוד בכל הדרישות. עוד דבר שהזמן משפיע עליו הוא ספר הפרויקט. למרות שאני כן חושב שספר הפרויקט ברמה גבוהה, הייתי שמח אם היה לי עוד זמן להשקיע בו יותר ולדאוג שהתיאור יהיה ברור יותר ומקיף יותר.

**תיאור תהליך הלמידה**

ניתן למצוא תיאור כולל של תהליך הלמידה [**ביומן השיקוף**](https://docs.google.com/document/d/1colcZbFaTTgTnlCAEUxlg_xhYFEFm8Mql9mpqI0yY3s/edit)**.** הוא נמצא גם כאן וגם בנספחים. אשמח אם תעבו עליו בפרוף מתוארים בו הרבה התמודויות אם קשיים,הצלחות, דרכי פתרון ובעיקר פשוט תהליך העבודה והלמידה בפרויקט. בנוסף מתוארים שם גם תבונות מדברים שלמדתי.

**בראיה לאחור:**בראיה לאחור הייתי מנסה לדחוף יותר את הפרויקט גם אם זה אומר לוותר קצת על למידה למבחנים אחרים. בסוף אני נורא מתבאס שאני לא מספיק לסיים את כל הפיצ'רים שתכננתי להוסיף וגם לא נעים לעשות את הספר פרויקט בדקה האחרונה עם הלשון בחוץ.

**במידה והיו ברשותי משאבים נוספים:** המשאב היחיד שאני צריך הוא זמן :) אני חושב שיש לי את היכולת להשיג את כל המטרות שהצבתי לעצמי בפרויקט רק חסר לי מעט זמן.

**תודות:**

אני רוצה להודות לאלי גולדשטיין. שתמך, עזר, סכרן ודחף אותי ואת שאר התלמידים ללמידה בצורה הכי טובה שיש! תמיד עודד לעצמאיות וניסה לגרום לנו להראות את ה100 אחוז שלנו. הוא אף פעם לא הסתפק בפרויקט ש"עובד" תמיד היה צריך להיות "מעולה". תודה רבה אלי. מעריך מאוד.

# ביבליוגרפיה

במהלך הפרויקט נעזרתי בכל מני מקומות שונים עבור כל מני דברים שונים הנה קישורים למקורות אלו:  
העברת קובץ בפייטון על גבי סוקט - [קישור](https://www.youtube.com/watch?v=qFVoMo6OMsQ)

חקירה על message object של דיסקורד - [קישור](https://discord.com/developers/docs/resources/channel#message-object)

חקירה על attachment object של דיסקורד - [קישור](https://discord.com/developers/docs/resources/channel#attachment-object)

שאלה שעזרה לי כשנתקעתי בפרויקט - [קישור](https://stackoverflow.com/questions/52232177/runtimeerror-timeout-context-manager-should-be-used-inside-a-task)

מחקר על sqlite types - [קישור](https://www.sqlite.org/datatype3.html)

למידה על slite3 עבור python - [קישור](https://www.youtube.com/watch?v=pd-0G0MigUA)

# נספחים

אם ירדתם לפה עבור הקוד של המימושים, תכף נגיע לזה. לפני זה הנה כמה דברים אחרים שחשוב לי לשתף פה לפני שאני מבלגן הכל עם קוד :)

**מצגת הצעת פרויקט**: מצגת ראשונית שערכתי עבור הצעת הרעיון לאלי.



**מצגת עדכון פרויקט:** אלי ביקש עדכון על מהלך העבודה אז הכנתי את המצגת הזאת:



**יומן שיקוף**: יומן שיקוף מתאר כמעט את כל תהליך הלמידה שלי וניתן למצוא קישור אליו כאן

<https://docs.google.com/document/d/1colcZbFaTTgTnlCAEUxlg_xhYFEFm8Mql9mpqI0yY3s/edit>

אני מקווה בעתיד לעדכן אותו ככה שיכיל את כל תהליך הלמידה מהתחלה ועד הסוף.

**Discord API –** קישור לAPI שבו השתמשתי עבור הבוט - [ניתן למצוא פה.](https://discord.com/developers/docs/resources/application) בצד שמאל של האתר יש תפריט שמאוד נוח להשתמש בו.

**קישור לסרטון.** כמו שאמרתי במהלך הפרויקט מצאתי מישהו שעשה דבר דומה למה שאני עושה ניתן למצוא קישור לסרטון שלו [כאן.](https://www.youtube.com/watch?v=c_arQ-6ElYI&t=201s)

**BOT TOKEN -**REMOVED..\_

**ועכשיו מה שכולם חיכו לו. קוד הפרויקט.**

את כל קוד הפרויקט ניתן למצוא בעמוד [GItHub](https://github.com/IthamarBaron/DICE) . לספר אני מצרף גרסה ישנה יותר (שחסרים בה חלק מהצפנות למרות שהם כבר עובדות בגלל שהם עוד לא הוכנסו במאה אחוז לפרויקט)

**Client:**

import base64

import os

import json

import time

import socket

import Protocol

class Client:

def \_\_init\_\_(self, host, port):

self.symmetric\_key = None

self.asymmetric\_protocol\_instance = Protocol.AsymmetricEncryptionProtocol()

self.host = host

self.port = port

self.temp\_start\_time = None

self.temp\_end\_time = None

self.user\_data = None

self.users\_files = {}

self.reload\_files\_flag = False

self.packet\_handlers = [self.handle\_server\_key, self.handle\_login\_data, self.get\_file\_from\_server,

self.handle\_files\_for\_initiation]

def connect\_to\_server(self) -> bool:

"""

Connect to the server at the specified host and port.

This method creates a client socket, connects it to the server, and prints a success

message upon successful connection.

:return: Bool

"""

try:

self.client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.client\_socket.connect((self.host, self.port))

print(f"Connected to the server at {self.host}:{self.port}")

Protocol.Protocol.load\_key()

return True

except Exception as e:

print(f"Connection error: {str(e)}")

return False

def receive\_data(self):

try:

print("receive data called!")

packet\_id = int(self.client\_socket.recv(1).decode())

print(f"packetID {packet\_id}")

data\_length = int(self.client\_socket.recv(4).decode())

print(f"data\_length {data\_length}")

print(f"calling {str(self.packet\_handlers[packet\_id])}")

self.packet\_handlers[packet\_id](data\_length)

except ValueError as e:

# added this part to avoid a small error blocking new attempts

print(f"ValueError - {e}. trying to clean the socket")

self.client\_socket.recv(1240)

except Exception as e:

print(f"Error receiving data: {e}")

raise e

def get\_file\_from\_server(self, data\_length):

try:

file\_info = self.client\_socket.recv(int(data\_length)).decode()

file\_info = json.loads(file\_info)

file\_name = file\_info["file\_name"]

file\_size = int(file\_info["len\_file\_bytes"])

print(f"Receiving file: {file\_name} of size: {file\_size / (1024 \*\* 2):.2f} MB")

received\_bytes\_count = 0

file = open(f"received\_{file\_name}", 'wb')

while received\_bytes\_count != file\_size:

part\_of\_file = self.client\_socket.recv(file\_size - received\_bytes\_count)

if not len(part\_of\_file):

print("Connection lost")

# TODO: do something about this (make faster)

file.write(part\_of\_file)

received\_bytes\_count += len(part\_of\_file)

file.close()

self.temp\_end\_time = time.time()

delta\_time = int(self.temp\_end\_time - self.temp\_start\_time)

print(f"time time elapsed: {delta\_time}")

print(f"Download average speed: {(file\_size / (1024 \*\* 2)) / delta\_time}")

except Exception as e:

print(f"Error receive file: {e}")

def send\_file\_to\_server(self, file\_path: str, channel\_id: int) -> None:

"""

Send a file to the server.

This method sends the file name, file size, and file data to the server

:param file\_path: The path of the file to send

:param channel\_id: channel that the message will be stored in

"""

file\_name = os.path.basename(file\_path)

file = open(file\_path, "rb")

file\_size = os.path.getsize(file\_path)

file\_data = file.read()

file.close()

packet = {

"file\_name": file\_name,

"file\_size": file\_size,

"channel\_id": channel\_id

}

print(f"FILEDATA {packet}")

file\_info = Protocol.Protocol.prepare\_file\_info\_to\_send(packet, packet\_id=2)

#data\_to\_send = f"{2}{Client.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.client\_socket.send(file\_info)

self.client\_socket.sendall(file\_data)

self.reload\_files\_flag = True

def request\_signup(self, username, password):

packet = {

"username": username,

"password": password

}

data\_to\_send = Protocol.Protocol.prepare\_data\_to\_send(1, packet)

#data\_to\_send = f"{1}{Client.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.client\_socket.send(data\_to\_send)

response = self.client\_socket.recv(19)

print(response)

if response == 0:

return False

else:

return True

def request\_file\_deletion(self, file\_name, channel\_id):

packet = {

"file\_name": file\_name,

"channel\_id": channel\_id

}

print("method called")

data\_to\_send = Protocol.Protocol.prepare\_data\_to\_send(4, packet)

#data\_to\_send = f"{4}{Client.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.client\_socket.send(data\_to\_send)

self.reload\_files\_flag = True

print("Sent file request")

def request\_download\_file(self, file\_name, channel\_id):

self.temp\_start\_time = time.time()

packet = {

"file\_name": file\_name,

"channel\_id": channel\_id

}

data\_to\_send = Protocol.Protocol.prepare\_data\_to\_send(3,packet)

#data\_to\_send = f"{3}{Client.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.client\_socket.send(data\_to\_send)

self.receive\_data()

self.reload\_files\_flag = True

def request\_login(self, username, password):

packet = {

"username": username,

"password": password,

}

data\_to\_send = Protocol.Protocol.prepare\_data\_to\_send(5, packet)

#data\_to\_send = f"{5}{Client.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.client\_socket.send(data\_to\_send)

self.receive\_data()

def request\_user\_files(self):

packet = {

"channel\_id": self.user\_data[2]

}

data\_to\_send = Protocol.Protocol.prepare\_data\_to\_send(6,packet)

#data\_to\_send = f"{6}{self.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.client\_socket.send(data\_to\_send)

self.receive\_data()

def handle\_files\_for\_initiation(self, data\_length):

data = self.client\_socket.recv(data\_length).decode()

data = json.loads(data)

self.users\_files = data["files"]

self.reload\_files\_flag = True

def handle\_login\_data(self, data\_length):

data = self.client\_socket.recv(data\_length).decode()

data = json.loads(data)

print(data["row"])

self.user\_data = data["row"]

def handle\_server\_key(self, data\_length):

data = self.client\_socket.recv(data\_length).decode()

data = json.loads(data)

self.server\_publc\_key = self.asymmetric\_protocol\_instance.load\_public\_key\_from\_str(data["server\_public\_key"])

print(f"Server public key in client {self.server\_publc\_key}")

# Generate a symmetric key:

if not self.symmetric\_key:

self.symmetric\_key = Protocol.get\_random\_bytes(16)

print(f"type of shit {type(self.server\_publc\_key)}")

encrypted\_symmetric\_key = self.asymmetric\_protocol\_instance.encrypt\_symmetric\_key(self.symmetric\_key, self.server\_publc\_key)

encrypted\_symmetric\_key\_base64 = base64.b64encode(encrypted\_symmetric\_key).decode('utf-8')

#TODO: SEND THE SYMMETRIC KEY

packet = {

"encrypted\_symmetric\_key\_base64": encrypted\_symmetric\_key\_base64,

}

data\_to\_send = Protocol.Protocol.prepare\_data\_to\_send(0, packet)

self.client\_socket.sendall(data\_to\_send)

@staticmethod

def zero\_fill\_length(input\_string, width=4):

length = len(input\_string)

length\_str = str(length).zfill(width)

return length\_str

"""if \_\_name\_\_ == "\_\_main\_\_":

client = Client('LocalHost', 12345)

client.connect\_to\_server()

while True:

client.receive\_data()"""

**ClientUI**

import os

import threading

import tkinter as tk

from tkinter import filedialog

from tkinter import PhotoImage

from Client import Client

from tkinter import font

BACKGROUND\_COLOR = "#5f8ac2"

class ManagerUI:

def \_\_init\_\_(self, root):

self.root = root

self.current\_frame = None

self.client\_instance = None # type: Client

self.connect\_frame()

self.display\_files = []

# region FRAMES

def connect\_frame(self):

if self.current\_frame:

self.current\_frame.destroy()

self.current\_frame = tk.Frame(self.root, bg=BACKGROUND\_COLOR)

self.current\_frame.pack(expand=True, fill=tk.BOTH)

self.root.geometry("500x400")

self.root.title("Connect to Server")

# title

large\_text\_label = tk.Label(self.current\_frame, text="CONNECT TO SERVER", font=("Arial", 24, "bold"),

bg=BACKGROUND\_COLOR, fg="white")

large\_text\_label.pack(pady=20)

# photo

image\_path = "images/server\_connection\_icon.png"

self.image = PhotoImage(master=self.current\_frame, width=200, height=100, file=image\_path)

image\_label = tk.Label(self.current\_frame, image=self.image, bg=BACKGROUND\_COLOR)

image\_label.pack(pady=(20, 0))

# String Input Field

self.entry = tk.Entry(self.current\_frame, font=('Arial', 14), fg="white", bg="#D3D3D3", bd=3, relief="solid",

insertbackground="black")

self.entry.insert(0, "127.0.0.1")

self.entry.pack(pady=20)

self.error\_label\_0 = tk.Label(self.current\_frame, text="", font=("Arial", 12, "bold"), fg="#cf3e3e",

bg=BACKGROUND\_COLOR)

self.error\_label\_0.pack()

# Connect Button

connect\_button = tk.Button(self.current\_frame, text="Connect", command=self.connect\_to\_server, bg="#4CAF50",

fg="black",

bd=2, relief="solid", width=15, height=2)

connect\_button.pack(pady=20)

# labels

label\_dice = tk.Label(self.current\_frame, text="DICE", font=("Rubik", 10, "bold", "underline"), fg="white",

bg=BACKGROUND\_COLOR)

label\_description = tk.Label(self.current\_frame, text="~Discord Integrated Cryptographic Engine",

font=("Rubik", 9, "bold"), fg="white", bg=BACKGROUND\_COLOR)

label\_description.pack(side=tk.BOTTOM, anchor=tk.W)

label\_dice.pack(side=tk.BOTTOM, anchor=tk.W)

def login\_signup\_frame(self):

if self.current\_frame:

self.current\_frame.destroy()

self.current\_frame = tk.Frame(self.root, bg=BACKGROUND\_COLOR)

self.current\_frame.pack(expand=True, fill=tk.BOTH)

self.root.geometry("500x400")

# Title

title\_label = tk.Label(self.current\_frame, text="LOG IN/SIGN UP", font=("Rubik", 25, "bold"),

bg=BACKGROUND\_COLOR, fg="white")

title\_label.pack(side=tk.TOP, pady=20)

# Username Section

username\_label = tk.Label(self.current\_frame, text="Name", font=("Arial", 12, "bold"), bg=BACKGROUND\_COLOR,

fg="black")

username\_label.pack(side=tk.TOP, padx=20, pady=(0, 0))

self.username\_entry = tk.Entry(self.current\_frame, bg="#D3D3D3", fg="#1f1f1f", bd=2, insertbackground="black",

width=45

, relief=tk.SOLID, )

self.username\_entry.insert(0, "Enter username")

self.username\_entry.pack(side=tk.TOP, padx=20, pady=(0, 10), ipady=5)

# Password Section

password\_label = tk.Label(self.current\_frame, text="Password", font=("Arial", 12, "bold"), bg=BACKGROUND\_COLOR,

fg="black")

password\_label.pack(side=tk.TOP, padx=20, pady=(0, 0))

self.password\_entry = tk.Entry(self.current\_frame, show="\*", bg="#D3D3D3", fg="#1f1f1f", bd=2,

insertbackground="black",

width=45, relief=tk.SOLID)

self.password\_entry.pack(side=tk.TOP, padx=20, pady=(0, 10), ipady=5)

self.error\_label = tk.Label(self.current\_frame, text="", font=("Arial", 12, "bold"), fg="#cf3e3e",

bg=BACKGROUND\_COLOR)

self.error\_label.pack()

# Buttons

signup\_button = tk.Button(self.current\_frame, text="Sign Up", bg="#14427d", fg="white", bd=3,

command=self.signup, relief=tk.SOLID, width=10, height=2)

signup\_button.place(relx=0.28, rely=0.6, anchor=tk.CENTER)

login\_button = tk.Button(self.current\_frame, text="Login", bg="#08a64f", fg="white", bd=3,

relief=tk.SOLID, command=self.login, width=10, height=2)

login\_button.place(relx=.72, rely=0.6, anchor=tk.CENTER)

# labels

label\_dice = tk.Label(self.current\_frame, text="DICE", font=("Rubik", 10, "bold", "underline"), fg="white",

bg=BACKGROUND\_COLOR)

label\_description = tk.Label(self.current\_frame, text="~Discord Integrated Cryptographic Engine",

font=("Rubik", 9, "bold"), fg="white", bg=BACKGROUND\_COLOR)

label\_description.pack(side=tk.BOTTOM, anchor=tk.W)

label\_dice.pack(side=tk.BOTTOM, anchor=tk.W)

def main\_application\_frame(self):

self.initiate\_users\_files()

self.current\_frame = tk.Frame(self.root, bg=BACKGROUND\_COLOR)

self.root.title("File Upload")

# title

self.title\_label = tk.Label(self.root, text="DICE - Debug panel", font=("Arial", 16))

self.title\_label.pack(pady=10)

# file upload button

self.upload\_button = tk.Button(self.root, text="Upload File", command=self.upload\_file)

self.upload\_button.pack(pady=20)

# send button

self.send\_button = tk.Button(self.root, text="Send", command=self.send\_data)

self.send\_button.pack(pady=20)

# reload button

self.reload\_files\_button = tk.Button(self.root, text="Refresh Files", command=self.reload\_files)

self.reload\_files\_button.pack(anchor="se")

self.reload\_files\_buttont = tk.Button(self.root, text="CLEAR Files", command=self.clear\_file\_labels)

self.reload\_files\_buttont.pack(anchor="se")

self.reload\_files\_buttontt = tk.Button(self.root, text="INNIT Files", command=self.initiate\_users\_files)

self.reload\_files\_buttontt.pack(anchor="se")

self.reload\_files\_buttontt = tk.Button(self.root, text="INSTANTIATE Files",

command=self.instantiate\_file\_labels)

self.reload\_files\_buttontt.pack(anchor="se")

self.instantiate\_file\_labels()

self.root.after(200, self.check\_reload\_flag)

# endregion FRAMES

# region COMMANDS

def connect\_to\_server(self):

ip = self.entry.get()

print(f"Attempting to connect to: {ip}")

self.client\_instance = Client(ip, 12345)

is\_successful = self.client\_instance.connect\_to\_server()

if not is\_successful:

self.error\_label\_0.config(text="Connection failed, Try again")

return

self.client\_instance.receive\_data()

# If connection is successful, switch to the file upload frame

self.current\_frame.destroy()

self.login\_signup\_frame()

def upload\_file(self):

self.uploded\_file\_path = filedialog.askopenfilename()

file\_size = os.path.getsize(self.uploded\_file\_path)

file\_size = file\_size / (1024 \*\* 3) # gb

if file\_size > 1:

self.uploded\_file\_path = None

# TODO: DISPLAY FILE LARGER ERROR

return

def send\_data(self):

try:

if self.uploded\_file\_path:

thread = threading.Thread(target=self.client\_instance.send\_file\_to\_server,

args=(self.uploded\_file\_path, self.client\_instance.user\_data[2]), daemon=True)

thread.start()

print("Finished sending method!")

except Exception as e:

print(f"Error sending the file to the server: {e}")

def login(self):

username = self.username\_entry.get()

password = self.password\_entry.get()

print(f"attempting login as {username} with {password}")

self.client\_instance.request\_login(username, password)

login\_data = self.client\_instance.user\_data

if login\_data:

self.client\_instance.user\_data = login\_data

# if login is successful we move on to the app

self.current\_frame.destroy()

self.main\_application\_frame()

else:

self.error\_label.config(text="Login Failed\ninvalid credentials")

pass

def signup(self):

username = self.username\_entry.get()

password = self.password\_entry.get()

if len(username) == 0 or len(password) == 0:

self.error\_label.config(text="SignUp Failed\nFill both fields")

return

is\_successful = self.client\_instance.request\_signup(username, password)

if is\_successful:

self.login()

else:

self.error\_label.config(text="SignUp Failed\nName Is Taken")

def download\_file(self, filename):

thread = threading.Thread(target=self.client\_instance.request\_download\_file,

args=(filename, int(self.client\_instance.user\_data[2])))

thread.start()

def reload\_files(self):

# Clear existing labels and buttons

self.clear\_file\_labels()

# Update underlying data (self.client\_instance.users\_files)

self.initiate\_users\_files()

# Instantiate new labels and buttons using updated data

self.instantiate\_file\_labels()

def delete\_file(self, filename):

print(f"deleting file {filename}")

thread = threading.Thread(target=self.client\_instance.request\_file\_deletion,

args=(filename, int(self.client\_instance.user\_data[2])))

thread.start()

# region File-display

def check\_reload\_flag(self):

if self.client\_instance.reload\_files\_flag:

self.reload\_files()

self.client\_instance.reload\_files\_flag = False

self.root.after(20, self.check\_reload\_flag)

#TODO: ALSO MAKE THIS SERER SIDE

def initiate\_users\_files(self):

self.client\_instance.request\_user\_files()

files = self.client\_instance.users\_files

self.client\_instance.users\_files = {} # clearing the dict before instantiation

for file in files:

self.client\_instance.users\_files[file[0]] = file[1]

def instantiate\_file\_labels(self):

for file\_name in self.client\_instance.users\_files.keys():

# Frame to group components for each file

file\_frame = tk.Frame(self.root)

file\_frame.pack()

# File Name Label

file\_label = tk.Label(file\_frame, text=file\_name)

file\_label.pack(side=tk.LEFT, padx=5)

# Download Button

download\_button = tk.Button(file\_frame, text="Download", fg="green",

command=lambda name=file\_name: self.download\_file(name))

download\_button.pack(side=tk.LEFT, padx=5)

# Delete Button

delete\_button = tk.Button(file\_frame, text="Delete", fg="red",

command=lambda name=file\_name: self.delete\_file(name))

delete\_button.pack(side=tk.LEFT, padx=5)

self.display\_files.extend([file\_frame, file\_label, download\_button, delete\_button])

def clear\_file\_labels(self):

for button in self.display\_files:

button.destroy()

self.display\_files = []

# endregion File-display

# endregion COMMANDS

def main():

root = tk.Tk()

app\_controller = ManagerUI(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**DatabaseInitializer**

import sqlite3

def initialize\_database(db\_name):

conn = sqlite3.connect(db\_name)

cursor = conn.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS users (

username TEXT PRIMARY KEY,

password TEXT NOT NULL,

channel\_id TEXT NOT NULL

)

""")

conn.commit()

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

db\_name = 'Dice-Database.db'

initialize\_database(db\_name)

print(f"Database '{db\_name}' has been initialized with the necessary tables.")

**DatabaseManager**

import sqlite3

class Database:

def \_\_init\_\_(self, db\_name):

self.db\_name = db\_name

self.conn = sqlite3.connect(db\_name)

self.cursor = self.conn.cursor()

def attempt\_login(self, username, password):

try:

self.cursor.execute(f"SELECT \* FROM users WHERE username='{username}' AND password='{password}'")

row = self.cursor.fetchone()

self.conn.commit()

return row

except Exception as e:

print(f"Error during login: {e} [db\_name: {self.db\_name}]")

def is\_username\_availability(self,username):

self.cursor.execute(f"SELECT \* FROM users WHERE username='{username}'")

row = self.cursor.fetchone()

self.conn.commit()

if row:

return False

return True

def create\_new\_account(self, username, password, channel\_id):

try:

self.cursor.execute(f"INSERT INTO users VALUES ('{username}', '{password}', '{channel\_id}')")

self.create\_new\_file\_table(channel\_id)

except Exception as e:

print(f"Error creating a new account: {e}")

finally:

self.conn.commit()

def create\_new\_file\_table(self, channel\_id):

try:

self.cursor.execute(f"""CREATE TABLE IF NOT EXISTS \_{channel\_id} (

filename TEXT,

message\_id INTEGER

)""")

except Exception as e:

print(f"Error during new table creation: {e}")

pass

def new\_file\_in\_channel(self, filename, message\_id, channel\_id):

try:

self.cursor.execute(f"INSERT INTO \_{channel\_id} VALUES ('{filename}', '{message\_id}')")

except Exception as e:

print(f"Error inserting a new file: {e}")

finally:

self.conn.commit()

def get\_files\_from\_id(self, channel\_id):

try:

self.cursor.execute(f"SELECT \* FROM \_{channel\_id}")

files = self.cursor.fetchall()

self.conn.commit()

return files

except Exception as e:

print(f"Error getting files from id: {e}")

self.conn.commit()

def get\_message\_id\_by\_name(self, filename, channel\_id):

try:

self.cursor.execute(f"SELECT \* FROM \_{channel\_id} WHERE filename='{filename}'")

info = self.cursor.fetchone()

message\_id = info[1]

self.conn.commit()

return int(message\_id)

except Exception as e:

print(f"Error getting messageID: {e}")

self.conn.commit()

return 0

def delete\_file\_in\_table(self, filename, channel\_id):

try:

self.cursor.execute(f"DELETE FROM \_{channel\_id} WHERE filename='{filename}'")

self.conn.commit()

except Exception as e:

print(f"Error deleting file {e}")

**DiscordBot**  
import io

import re

import time

import discord

import requests

from FileManager import FileManager

from discord import Activity, ActivityType

class DiscordBot:

def \_\_init\_\_(self, token):

self.token = token

self.intents = discord.Intents.all()

self.intents.typing = False

self.intents.presences = False

self.intents.messages = True

self.intents.message\_content = True

self.bot = discord.Client(intents=self.intents)

self.file\_manager = FileManager()

async def get\_available\_file\_name(self, file\_name, channel\_id=1182998507460771890):

channel = self.bot.get\_channel(channel\_id)

all\_messages = set()

async for msg in channel.history(limit=None):

if not msg.reference: # Check if the message is not a reply

all\_messages.add(msg.content)

found\_valid\_name = False

i = 1

available\_name = file\_name

while not found\_valid\_name:

if available\_name in all\_messages:

available\_name = file\_name+f"({i})"

i += 1

else:

found\_valid\_name = True

return available\_name

async def create\_new\_storage\_area(self,channel\_name):

guild\_id = 1182329387115348039

guild = self.bot.get\_guild(guild\_id)

category = discord.utils.get(guild.categories, id=1182329387887104081)

new\_channel = await guild.create\_text\_channel(channel\_name, category=category)

return new\_channel.id

async def get\_message\_id\_by\_content(self, channel\_name: str, message\_content: str) -> int:

guild = self.bot.guilds[0]

channel = discord.utils.get(guild.channels, name=channel\_name)

if channel:

async for message in channel.history(limit=100):

if message.content == message\_content:

return message.id

return -1 # message not found

async def on\_ready(self):

print(f"{self.bot.user} is running")

activity = Activity(name="Information", type=ActivityType.watching)

await self.bot.change\_presence(activity=activity)

async def send\_file\_in\_chat(self, file\_name, file\_content, channel\_id=1182998507460771890):

try:

file\_name = await self.get\_available\_file\_name(file\_name, channel\_id)

channel = self.bot.get\_channel(channel\_id)

await channel.send("File sending in process.")

await channel.send(file\_name)

message\_id = await self.get\_message\_id\_by\_content(str(channel), file\_name)

reference\_message = await channel.fetch\_message(message\_id)

data\_list = self.file\_manager.split\_file\_data(file\_content)

for i, chunk in enumerate(data\_list):

file\_data = discord.File(chunk, filename=f"{reference\_message.content}{i}.txt")

await channel.send(content=f"Chunk {i + 1}:", file=file\_data, reference=reference\_message)

await channel.send("File sending complete.")

return [file\_name, message\_id]

except Exception as e:

print(f"Error sending file in chat: {e}")

return None

async def assemble\_file\_from\_chat(self, message\_id, channel\_id=1182998507460771890):

try:

if isinstance(channel\_id, int):

channel = self.bot.get\_channel(channel\_id)

else:

channel = self.bot.get\_channel(int(channel\_id))

await channel.send("File assembly in progress...")

reference\_message = await channel.fetch\_message(message\_id)

all\_messages = []

async for msg in reference\_message.channel.history(after=reference\_message, limit=20):

all\_messages.append(msg)

reply\_messages = [msg for msg in all\_messages if msg.reference and msg.reference.message\_id == message\_id]

sorted\_reply\_messages = sorted(reply\_messages, key=lambda msg: msg.created\_at)

print(f"Type of sorted: {type(sorted\_reply\_messages)} type of [0] {type(sorted\_reply\_messages[0])}")

file\_bytes = self.file\_manager.assemble\_file(sorted\_reply\_messages)

await channel.send("File assembly completed!")

return file\_bytes

except Exception as e:

await channel.send("File assembly failed!")

print(f"Error during file assembling: {e}")

return b""

async def delete\_file\_from\_chat(self, message\_id, channel\_id=1182998507460771890):

try:

if isinstance(channel\_id, int):

channel = self.bot.get\_channel(channel\_id)

else:

channel = self.bot.get\_channel(int(channel\_id))

reference\_message = await channel.fetch\_message(message\_id)

await channel.send(f"Deleting File: {reference\_message.content}")

all\_messages = []

async for msg in reference\_message.channel.history(after=reference\_message, limit=20):

all\_messages.append(msg)

reply\_messages = [msg for msg in all\_messages if msg.reference and msg.reference.message\_id == message\_id]

for msg in reply\_messages:

await msg.delete()

await reference\_message.delete()

print(f"Deletion process completed")

return True

except Exception as e:

return False

async def on\_message(self, message: discord.Message):

if message.author == self.bot.user:

return # Don't respond to ourselves

username = str(message.author)

user\_message = str(message.content)

channel = message.channel

print(f"{username} said: {user\_message} in {channel} attachments {message.attachments}")

await message.channel.send(f"Noticed Input")

# message command

if user\_message.lower().startswith("send"): # send <filename> # TODO: properly make filename and name

name = (user\_message[5::])

file\_name = name

file = open(file\_name, "rb")

file\_data = file.read()

file.close()

await self.send\_file\_in\_chat(name, file\_data)

elif user\_message.lower().startswith("delete"):

message\_params = user\_message.split(" ")

await self.delete\_file\_from\_chat(int(message\_params[1]), int(message\_params[2]))

print("deleting")

elif user\_message.lower().startswith("log"):

pass

elif user\_message.lower().startswith("clear"):

all\_messages = []

async for msg in message.channel.history(limit=int(user\_message[6::])):

all\_messages.append(msg)

for msg in all\_messages:

await msg.delete()

def run\_discord\_bot(self):

@self.bot.event

async def on\_ready():

await self.on\_ready()

@self.bot.event

async def on\_message(message):

await self.on\_message(message)

self.bot.run(self.token)

"""

bot = DiscordBot("")

bot.run\_discord\_bot()"""

**Filemanager**

import io

import discord

import requests

from typing import List

class FileManager:

CHUNK = 25 \* 1024 \*\* 2 # 25 MB

def \_\_init\_\_(self):

pass

def split\_file\_data(self, file\_data: bytes) -> List[io.BytesIO]:

return [io.BytesIO(file\_data[pos: pos + self.CHUNK]) for pos in range(0, len(file\_data), self.CHUNK)]

def assemble\_file(self, sorted\_reply\_messages: List[discord.message.Message]):

file\_data = b""

for replay in sorted\_reply\_messages:

chunk\_link = replay.attachments[0].url

chunk\_data = requests.get(chunk\_link).content

file\_data += chunk\_data

return file\_data

**mainappui**

from pathlib import Path

# from tkinter import \*

# Explicit imports to satisfy Flake8

from tkinter import Tk, Canvas, Entry, Text, Button, PhotoImage

OUTPUT\_PATH = Path(\_\_file\_\_).parent

ASSETS\_PATH = OUTPUT\_PATH / Path(r"assets\frame0")

def relative\_to\_assets(path: str) -> Path:

return ASSETS\_PATH / Path(path)

window = Tk()

window.geometry("1090x720")

window.configure(bg="#5F9BC6")

canvas = Canvas(

window,

bg = "#5F9BC6",

height = 720,

width = 1090,

bd = 0,

highlightthickness = 0,

relief = "ridge"

)

canvas.place(x = 0, y = 0)

image\_image\_1 = PhotoImage(

file=relative\_to\_assets("image\_1.png"))

image\_1 = canvas.create\_image(

545.0,

27.0,

image=image\_image\_1

)

canvas.create\_rectangle(

0.0,

55.0,

1090.0,

66.0,

fill="#000000",

outline="")

canvas.create\_text(

59.0,

9.0,

anchor="nw",

text="Welcome To Dice - A Discord Integrated Cryptographic Engine",

fill="#FFFFFF",

font=("MontserratRoman ExtraBold", 27 \* -1)

)

canvas.create\_rectangle(

13.0,

159.0,

320.0,

162.0,

fill="#D9D9D9",

outline="")

canvas.create\_rectangle(

13.0,

222.0,

320.0,

225.0,

fill="#D9D9D9",

outline="")

canvas.create\_rectangle(

5.0,

320.0,

312.0,

323.0,

fill="#D9D9D9",

outline="")

canvas.create\_text(

18.0,

124.0,

anchor="nw",

text="Select File to Upload",

fill="#FFFFFF",

font=("MontserratRoman ExtraBold", 27 \* -1)

)

canvas.create\_text(

18.0,

187.0,

anchor="nw",

text="Upload Selected File",

fill="#FFFFFF",

font=("MontserratRoman ExtraBold", 27 \* -1)

)

canvas.create\_text(

10.0,

285.0,

anchor="nw",

text="Your Uploaded Files :",

fill="#FFFFFF",

font=("MontserratRoman ExtraBold", 27 \* -1)

)

button\_image\_1 = PhotoImage(

file=relative\_to\_assets("button\_1.png"))

button\_1 = Button(

image=button\_image\_1,

borderwidth=0,

highlightthickness=0,

command=lambda: print("button\_1 clicked"),

relief="flat",

activebackground="#3678B6",

activeforeground="#3678B6"

)

button\_1.place(

x=332.0,

y=125.0,

width=147.0,

height=37.0

)

button\_image\_2 = PhotoImage(

file=relative\_to\_assets("button\_2.png"))

button\_2 = Button(

image=button\_image\_2,

borderwidth=0,

highlightthickness=0,

command=lambda: print("button\_2 clicked"),

relief="flat",

activebackground="#3678B6",

activeforeground="#3678B6"

)

button\_2.place(

x=332.0,

y=188.0,

width=147.0,

height=37.0

)

button\_image\_5 = PhotoImage(

file=relative\_to\_assets("button\_5.png"))

button\_5 = Button(

image=button\_image\_5,

borderwidth=0,

highlightthickness=0,

command=lambda: print("button\_5 clicked"),

relief="flat",

activebackground="#3678B6",

activeforeground="#3678B6"

)

button\_5.place(

x=477.0,

y=388.0,

width=148.0,

height=37.0

)

button\_image\_8 = PhotoImage(

file=relative\_to\_assets("button\_8.png"))

button\_8 = Button(

image=button\_image\_8,

borderwidth=0,

highlightthickness=0,

command=lambda: print("button\_8 clicked"),

relief="flat",

activebackground="#3678B6",

activeforeground="#3678B6"

)

button\_8.place(

x=315.0,

y=388.0,

width=149.0,

height=37.0

)

image\_image\_2 = PhotoImage(

file=relative\_to\_assets("image\_2.png"))

image\_2 = canvas.create\_image(

33.0,

23.0,

image=image\_image\_2

)

image\_image\_3 = PhotoImage(

file=relative\_to\_assets("image\_3.png"))

image\_3 = canvas.create\_image(

78.0,

406.0,

image=image\_image\_3

)

canvas.create\_text(

123.0,

392.0,

anchor="nw",

text="File Name.txt",

fill="#626262",

font=("MontserratRoman SemiBold", 24 \* -1)

)

window.resizable(False, False)

window.mainloop()

**Protocol**

import json

import Crypto.Random

from cryptography.fernet import Fernet

import json

from cryptography.fernet import Fernet

from cryptography.hazmat.backends import default\_backend

from cryptography.hazmat.primitives import serialization, hashes

from cryptography.hazmat.primitives.asymmetric import rsa, padding

from Crypto.Cipher import AES

from Crypto.Util.Padding import pad, unpad

from Crypto.Random import get\_random\_bytes

class Protocol:

key = None

fernet\_object = None

@staticmethod

def generate\_key():

Protocol.key = Fernet.generate\_key()

Protocol.fernet\_object = Fernet(key=Protocol.key)

Protocol.debug\_protocol()

# Save the key to a file

with open("encryption\_key.txt", "wb") as key\_file:

key\_file.write(Protocol.key)

@staticmethod

def debug\_protocol():

"""

serves no purpose other than debugging

"""

print(f"============[PROTOCOL DEBUG]============")

print(f"[PROTOCOL KEY = {Protocol.key}]")

print(f"[PROTOCOL TYPE OF FERNET-OBJ = {type(Protocol.fernet\_object)}]")

print(f"============[PROTOCOL DEBUG]============")

@staticmethod

def load\_key():

# Load the key from the file

with open("encryption\_key.txt", "rb") as key\_file:

Protocol.key = key\_file.read()

Protocol.fernet\_object = Fernet(key=Protocol.key)

Protocol.debug\_protocol()

@staticmethod

def zero\_fill\_length(input\_string, width=4):

"""

Adds zeros in the beginning of the packet length field

"""

length = len(input\_string)

length\_str = str(length).zfill(width)

return length\_str

@staticmethod

def prepare\_data\_to\_send(packet\_id, packet):

"""

responsible for preparing "regular" data for sending to the server (not for files)

:param packet\_id: packet id for handling purposes on the receiving side

:param packet: NOT!! serialized packet dict

:type packet: dict

:return: data for sending

"""

data\_to\_encrypt = json.dumps(packet).encode()

encrypted\_packet = Protocol.fernet\_object.encrypt(data\_to\_encrypt)

encrypted\_packet\_str = encrypted\_packet.decode() # Convert bytes to string

packet\_length = Protocol.zero\_fill\_length(encrypted\_packet\_str) # Calculate packet length

data\_to\_send = f"{packet\_id}{packet\_length}{encrypted\_packet\_str}".encode()

return data\_to\_send

@staticmethod

def prepare\_file\_info\_to\_send(packet, packet\_id=2):

"""

responsible for preparing file data for sending to the server

:param packet\_id: packet id for handling purposes on the receiving side

:param packet: NOT!! serialized packet dict

:type packet: dict

:return: file data for sending

"""

data\_to\_encrypt = json.dumps(packet).encode()

encrypted\_packet = Protocol.fernet\_object.encrypt(data\_to\_encrypt)

encrypted\_packet\_str = encrypted\_packet.decode() # Convert bytes to string

packet\_length = Protocol.zero\_fill\_length(encrypted\_packet\_str) # Calculate packet length

data\_to\_send = f"{packet\_id}{packet\_length}{encrypted\_packet\_str}".encode()

return data\_to\_send

@staticmethod

def decrypt\_incoming\_data(encrypted\_data):

"""

decrypts incoming data using the fernet key

:param encrypted\_data: data to decrypt

:return: decrypted data

"""

decrypted\_data = None

try:

decrypted\_data = Protocol.fernet\_object.decrypt(encrypted\_data)

except Exception as e:

print(f"DECRYPTING ERROR {e}")

return decrypted\_data

class AsymmetricEncryptionProtocol:

def \_\_init\_\_(self):

"""

Initialize the ServerProtocol object with private and public key attributes set to None.

"""

self.private\_key = None

self.public\_key = None

@staticmethod

def get\_public\_key\_as\_str(public\_key):

# Serialize the public key to PEM format

pem = public\_key.public\_bytes(

encoding=serialization.Encoding.PEM,

format=serialization.PublicFormat.SubjectPublicKeyInfo

)

return pem.decode('utf-8')

@staticmethod

def load\_public\_key\_from\_str(pem\_str):

# Deserialize the public key from PEM format

public\_key = serialization.load\_pem\_public\_key(

pem\_str.encode('utf-8'),

backend=default\_backend()

)

return public\_key

def create\_server\_keys(self):

"""

Generate RSA private and public keys and store them in the respective attributes.

Returns:

None

"""

if not self.public\_key and not self.private\_key:

self.private\_key = rsa.generate\_private\_key(

public\_exponent=65537,

key\_size=2048,

backend=default\_backend()

)

self.public\_key = self.private\_key.public\_key()

def get\_public\_key(self):

"""

Retrieve the public key of the server in PEM format.

Returns:

bytes: The public key in PEM format.

"""

return self.public\_key.public\_bytes(

encoding=serialization.Encoding.PEM,

format=serialization.PublicFormat.SubjectPublicKeyInfo

)

def decrypt\_data(self, encrypted\_data):

"""

Decrypt encrypted data using the server's private key.

Args:

encrypted\_data (bytes): The data to decrypt.

Returns:

bytes: The decrypted data.

"""

return self.private\_key.decrypt(

encrypted\_data,

padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None

)

)

def encrypt\_data(self, data):

"""

Encrypt data using the server's public key.

Args:

data (bytes): The data to encrypt.

Returns:

bytes: The encrypted data.

"""

return self.public\_key.encrypt(

data,

padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None

)

)

def encrypt\_symmetric\_key(self, symmetric\_key, server\_public\_key):

"""

Encrypt a symmetric key using the server's public key.

Args:

symmetric\_key (bytes): The symmetric key to encrypt.

server\_public\_key (object): The server's public key used for encryption.

Returns:

bytes: The encrypted symmetric key.

"""

return server\_public\_key.encrypt(

symmetric\_key,

padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None

)

)

class ClientKeyTransferProtocol:

def \_\_init\_\_(self):

"""

Initialize the ClientProtocol object with the symmetric key attribute set to None.

"""

self.symmetric\_key = Fernet.generate\_key()

self.fernet\_object = Fernet(key=Protocol.key)

def generate\_symmetric\_key(self):

"""

Generate a symmetric key using a chosen method like Fernet or any other suitable method.

Returns:

None

"""

# Generate your symmetric key here using Fernet or any other method

pass

def encrypt\_symmetric\_key(self, symmetric\_key, server\_public\_key):

"""

Encrypt a symmetric key using the server's public key.

Args:

symmetric\_key (bytes): The symmetric key to encrypt.

server\_public\_key (object): The server's public key used for encryption.

Returns:

bytes: The encrypted symmetric key.

"""

return server\_public\_key.encrypt(

symmetric\_key,

padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None

)

)

def decrypt\_data\_using\_symmetric(self, encrypted\_data):

"""

Decrypt encrypted data using the symmetric key.

Args:

encrypted\_data (bytes): The data to decrypt.

Returns:

bytes: The decrypted data.

"""

# Logic for decrypting data using the symmetric key

pass

def encrypt\_data\_using\_symmetric(self, encrypted\_data):

"""

Encrypt data using the symmetric key.

Args:

encrypted\_data (bytes): The data to encrypt.

Returns:

bytes: The encrypted data.

"""

# Logic for encrypting data using the symmetric key

pass

def encrypt\_data\_asymmetric(self, data, public\_key):

"""

Encrypt data asymmetrically using the provided public key.

Args:

data (bytes): The data to encrypt.

public\_key (object): The public key used for encryption.

Returns:

bytes: The encrypted data.

"""

return public\_key.encrypt(

data,

padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),

algorithm=hashes.SHA256(),

label=None

)

)

class SymmetricEncryptionProtocol:

# use get\_random\_bytes(16) for key

@staticmethod

def encrypt\_data(key, data):

# Generate a random IV (Initialization Vector)

iv = get\_random\_bytes(16)

# Create AES cipher

cipher = AES.new(key, AES.MODE\_CBC, iv)

# Pad the data to be multiple of block size

padded\_data = pad(data, AES.block\_size)

# Encrypt the data

encrypted\_data = cipher.encrypt(padded\_data)

# Return the IV and encrypted data

return iv + encrypted\_data

@staticmethod

def decrypt\_data(key, encrypted\_data):

# Extract the IV from the beginning of the encrypted data

iv = encrypted\_data[:16]

# Extract the actual encrypted data

actual\_encrypted\_data = encrypted\_data[16:]

# Create AES cipher

cipher = AES.new(key, AES.MODE\_CBC, iv)

# Decrypt the data

decrypted\_data = cipher.decrypt(actual\_encrypted\_data)

# Unpad the data

original\_data = unpad(decrypted\_data, AES.block\_size)

# Return the original data

return original\_data

asymmetric = AsymmetricEncryptionProtocol()

symmetric = SymmetricEncryptionProtocol()

asymmetric.create\_server\_keys()

public\_key = asymmetric.public\_key

print(type(public\_key))

data = "test"

data\_bytes = data.encode('utf-8')

print(data\_bytes)

encrypted\_data\_key = asymmetric.encrypt\_symmetric\_key(data\_bytes, public\_key)

encrypted\_data\_data = asymmetric.encrypt\_data(data\_bytes)

print(f"encrypted data key {encrypted\_data\_key}")

print(f"encrypted data data {encrypted\_data\_data}")

decrypted\_data\_key = asymmetric.decrypt\_data(encrypted\_data\_key)

decrypted\_data\_data = asymmetric.decrypt\_data(encrypted\_data\_data)

print(f"decrypted data key {decrypted\_data\_key.decode('utf-8')}")

print(f"decrypted data data {decrypted\_data\_data.decode('utf-8')}")

**Server**

import asyncio

import json

import base64

import Protocol

from DatabaseManager import Database

import socket

import DiscordBot

import threading

class Server:

def \_\_init\_\_(self, host, port, token):

self.asymmetric\_protocol\_instance = Protocol.AsymmetricEncryptionProtocol()

self.host = host

self.port = port

self.clients = {}

self.clients\_symmetric\_keys = []

self.client\_threads = []

self.packet\_handlers = [self.handle\_symmetric\_key, self.handle\_sign\_up\_request, self.handle\_file\_and\_send\_to\_discord,

self.handle\_file\_request, self.handle\_deletion\_request, self.handle\_login\_request,

self.handle\_files\_for\_initiation]

#self.database = Database("Dice-Database.db")

self.bot\_instance = DiscordBot.DiscordBot(token)

thread = threading.Thread(target=self.bot\_instance.run\_discord\_bot, daemon=True)

thread.start()

def start(self) -> None:

"""

Start server and waits for client connections.

Creates a server socket, binds to host and port, and listens for clients.

Upon connection, prints client information.

:return: None

"""

Protocol.Protocol.generate\_key() #TODO: SWAP LATER

self.asymmetric\_protocol\_instance.create\_server\_keys()

print(f" [MAIN THREAD] Server public key in server {self.asymmetric\_protocol\_instance.get\_public\_key()}")

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind((self.host, self.port))

connected\_clients = 0

while True:

server\_socket.listen()

print(f" [MAIN THREAD] Server is listening for connections")

client\_socket, \_ = server\_socket.accept() # ACCEPT CONNECTION

print(f" [MAIN THREAD] client connected: {client\_socket.getpeername()}")

self.clients[connected\_clients] = [client\_socket] # Add the client to the dict

new\_client\_thread = threading.Thread(target=self.run\_server\_for\_client, args=(connected\_clients,), daemon=True)

self.client\_threads.append(new\_client\_thread)

new\_client\_thread.start()

connected\_clients += 1

def run\_server\_for\_client(self, client\_id):

self.clients[client\_id].append(Database("Dice-Database.db"))

packet = {

"server\_public\_key": self.asymmetric\_protocol\_instance.get\_public\_key\_as\_str(self.asymmetric\_protocol\_instance.public\_key),

}

data\_to\_send = f"{0}{self.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.clients[client\_id][0].sendall(data\_to\_send)

is\_connected = True

while is\_connected:

is\_connected = self.receive\_data(client\_id)

def receive\_data(self, client\_id):

try:

print("RECEIVING DATA")

packet\_id = int(self.clients[client\_id][0].recv(1).decode())

print(f" [CLIENT\_THREAD {client\_id}] packet\_id {packet\_id}")

data\_length = int(self.clients[client\_id][0].recv(4).decode())

print(f" [CLIENT\_THREAD {client\_id}] data\_length {data\_length}")

self.packet\_handlers[packet\_id](data\_length, client\_id)

except ValueError:

return True # client disconnected

except Exception as e:

raise e

print(f" [CLIENT\_THREAD {client\_id}] Error receiving data: {e}")

finally:

return True

def receive\_file(self, data\_length, client\_id):

"""

Receive a file from a connected client.

This method expects the client to send the file name, file size, and file data.

:return: None

"""

try:

data = self.clients[client\_id][0].recv(int(data\_length)).decode()

data = Protocol.Protocol.decrypt\_incoming\_data(data)

file\_info = json.loads(data)

file\_name = file\_info["file\_name"]

file\_size = int(file\_info["file\_size"])

channel\_id = int(file\_info["channel\_id"])

print(f"Receiving file: {file\_name} of size: {file\_size / (1024 \*\* 2):.2f} MB")

file\_bytes = b""

while len(file\_bytes) != file\_size:

part\_of\_file = self.clients[client\_id][0].recv(file\_size - len(file\_bytes))

if not len(part\_of\_file):

print("Connection lost")

return [0, 0, 0]

file\_bytes += part\_of\_file

#file\_bytes = Protocol.Protocol.decrypt\_incoming\_data(file\_bytes)

return [file\_name, file\_bytes, channel\_id]

except Exception as e:

print(f"Error receive file: {e}")

return [0, 0, 0]

def send\_files\_to\_discord(self, file\_name, file\_content, channel\_id, client\_id):

print(f" [CLIENT\_THREAD {client\_id}] Sending file [{file\_name}] in channel [{self.bot\_instance.bot.get\_channel(channel\_id)}]")

temp = asyncio.run\_coroutine\_threadsafe(

self.bot\_instance.send\_file\_in\_chat(file\_name, file\_content, channel\_id), self.bot\_instance.bot.loop)

reference\_message\_info = temp.result()

if reference\_message\_info:

print(f" [CLIENT\_THREAD {client\_id}] Message sent successfully")

self.clients[client\_id][1].new\_file\_in\_channel(reference\_message\_info[0], reference\_message\_info[1], channel\_id)

# region Handlers

def handle\_file\_and\_send\_to\_discord(self, data\_length, client\_id):

file\_data = self.receive\_file(data\_length, client\_id) # fil name | file content | channelID

if file\_data[1]:

self.send\_files\_to\_discord(file\_data[0], file\_data[1], file\_data[2], client\_id)

def handle\_sign\_up\_request(self, data\_length, client\_id):

data = self.clients[client\_id][0].recv(int(data\_length)).decode()

data = Protocol.Protocol.decrypt\_incoming\_data(data)

data = json.loads(data)

print(data)

if not self.clients[client\_id][1].is\_username\_availability(data["username"]):

print(f" [CLIENT\_THREAD {client\_id}] Username unavailable")

# hahaha i am 19 chars long

data = "noooooooooooooooooo"

data = f"{data}"

self.clients[client\_id][0].send(data.encode())

pass

else:

print(f" [CLIENT\_THREAD {client\_id}] username available")

attempt\_channel\_creation = asyncio.run\_coroutine\_threadsafe(self.bot\_instance.create\_new\_storage\_area(data["username"]),

self.bot\_instance.bot.loop)

channel\_id = attempt\_channel\_creation.result()

self.clients[client\_id][1].create\_new\_account(data["username"], data["password"], channel\_id)

print(channel\_id)

data = f"{channel\_id}"

self.clients[client\_id][0].send(data.encode())

def handle\_file\_request(self, data\_length, client\_id):

requested\_file\_info = self.clients[client\_id][0].recv(int(data\_length)).decode()

requested\_file\_info = Protocol.Protocol.decrypt\_incoming\_data(requested\_file\_info)

requested\_file\_info = json.loads(requested\_file\_info) # file\_name , channel\_id

message\_id = self.clients[client\_id][1].get\_message\_id\_by\_name(requested\_file\_info["file\_name"],

int(requested\_file\_info["channel\_id"]))

if message\_id != 0:

temp = asyncio.run\_coroutine\_threadsafe(

self.bot\_instance.assemble\_file\_from\_chat(message\_id, int(requested\_file\_info["channel\_id"])),

self.bot\_instance.bot.loop)

file\_bytes = temp.result()

packet = {

"file\_name": requested\_file\_info["file\_name"],

"len\_file\_bytes": len(file\_bytes)

}

data = f"{2}{self.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.clients[client\_id][0].sendall(data)

self.clients[client\_id][0].sendall(file\_bytes)

else:

print(f" [CLIENT\_THREAD {client\_id}] message id is 0")

def handle\_deletion\_request(self, data\_length,client\_id):

data = self.clients[client\_id][0].recv(data\_length).decode()

data = Protocol.Protocol.decrypt\_incoming\_data(data)

data = json.loads(data)

message\_id = self.clients[client\_id][1].get\_message\_id\_by\_name(data["file\_name"], int(data["channel\_id"]))

proc = asyncio.run\_coroutine\_threadsafe(self.bot\_instance.delete\_file\_from\_chat(message\_id,

int(data["channel\_id"])), self.bot\_instance.bot.loop)

is\_successful = proc.result()

if is\_successful:

print(f" [CLIENT\_THREAD {client\_id}] successful!")

self.clients[client\_id][1].delete\_file\_in\_table(data["file\_name"], data["channel\_id"])

# TODO: make threaded

def handle\_login\_request(self, data\_length, client\_id):

encrypted\_packet = self.clients[client\_id][0].recv(data\_length)

data = Protocol.Protocol.decrypt\_incoming\_data(encrypted\_packet )

data = json.loads(data.decode())

row = self.clients[client\_id][1].attempt\_login(data["username"], data["password"])

packet = {

"row": row

}

print(row)

data\_to\_send = f"{1}{self.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.clients[client\_id][0].sendall(data\_to\_send)

def handle\_files\_for\_initiation(self, data\_length, client\_id):

encrypted\_packet = self.clients[client\_id][0].recv(data\_length)

data = Protocol.Protocol.decrypt\_incoming\_data(encrypted\_packet )

data = json.loads(data.decode())

files = self.clients[client\_id][1].get\_files\_from\_id(data["channel\_id"])

packet = {

"files": files

}

print(f"Files: {files}")

data\_to\_send = f"{3}{self.zero\_fill\_length(str(packet))}{json.dumps(packet)}".encode()

self.clients[client\_id][0].send(data\_to\_send)

def handle\_symmetric\_key(self, data\_length, client\_id):

encrypted\_packet = self.clients[client\_id][0].recv(data\_length)

data = Protocol.Protocol.decrypt\_incoming\_data(encrypted\_packet)

data = json.loads(data.decode())

encrypted\_symmetric\_key\_base64 = data["encrypted\_symmetric\_key\_base64"]

encrypted\_symmetric\_key = base64.b64decode(encrypted\_symmetric\_key\_base64)

symmetric\_key = self.asymmetric\_protocol\_instance.decrypt\_data(encrypted\_symmetric\_key)

print(f"DECRYPTED SYMMETRIC KEY: [{symmetric\_key}]")

self.clients\_symmetric\_keys.append(symmetric\_key)

print(self.clients\_symmetric\_keys)

# endregion Handlers

@staticmethod

def zero\_fill\_length(input\_string, width=4):

length = len(input\_string)

length\_str = str(length).zfill(width)

return length\_str

if \_\_name\_\_ == "\_\_main\_\_":

server = Server('LocalHost', 12345, "")

server.start()