תיכון ברנר פתח תקוה משרד החינוך

**עבודת גמר הנדסת תוכנה**

**הנושא: CloudAV**  
 **לוגו הפרויקט**



שם התלמיד: תומר בן שושן

תעודת זהות: 216301697

שם המנחה: אמיר דבי

שם החלופה: הגנת סייבר ומערכות הפעלה

**יוני 2024 תשפ"ד**

**תוכן עניינים**

[**1.** **מבוא** 4](#_Toc162339473)

[**1.1.** **ייזום** 4](#_Toc162339474)

[**תקציר הפרויקט** 4](#_Toc162339475)

[**הגדרת הלקוח** 4](#_Toc162339476)

[**יעדים ומטרות** 4](#_Toc162339477)

[**בעיות תועלות וחסרונות** 5](#_Toc162339478)

[**פתרונות קיימים** 5](#_Toc162339479)

[**טכנולוגיית הפרויקט** 6](#_Toc162339480)

[**תחומי הפרויקט** 6](#_Toc162339481)

[**1.2.** **אפיון המערכת** 7](#_Toc162339484)

[**תיאור הפרויקט** 7](#_Toc162339485)

[**מאפיינים ויכולות** 7](#_Toc162339486)

[**בדיקות המערכת** 8](#_Toc162339487)

[**תכנון הבדיקה:** 8](#_Toc162339488)

[**תכנון וניהול לוח זמנים** 9](#_Toc162339489)

[**ניהול סיכונים** 10](#_Toc162339490)

[**2.** **תיאור מפורט של המערכת** 11](#_Toc162339491)

[**2.1.** **פירוט וניתוח יכולות המערכת** 11](#_Toc162339493)

[**פירוט היכולות בצד שרת** 11](#_Toc162339494)

[**פירוט היכולות בצד לקוח** 12](#_Toc162339495)

[**3.** **ארכיטקטורה** 13](#_Toc162339496)

[**3.1.** **ארכיטקטורת המערכת** 13](#_Toc162339500)

[**תיאור החומרה** 13](#_Toc162339501)

[**סכימת בלוקים** 13](#_Toc162339502)

[**3.2.** **טכנולוגיית המערכת** 14](#_Toc162339503)

[**שפת תכנות** 14](#_Toc162339504)

[**מערכת הפעלה** 14](#_Toc162339505)

[**3.3.** **זרימת המידע במערכת** 14](#_Toc162339506)

[**3.4.** **אלגוריתמים עיקריים** 15](#_Toc162339513)

[**3.5.** **סביבת הפיתוח** 15](#_Toc162339521)

[**כלי פיתוח נדרשים** 15](#_Toc162339522)

[**סביבת הבדיקות** 15](#_Toc162339523)

[**3.6.** **פרוטוקול התקשורת** 16](#_Toc162339524)

[**3.7.** **מסכי המערכת/ממשק משתמש** 17](#_Toc162339534)

[**3.8.** **מסד הנתונים** 23](#_Toc162339545)

[**3.9.** **חולשות ואיומים** 25](#_Toc162339557)

[**4.** **מימוש הפרויקט** 26](#_Toc162339558)

[**4.1.** **מודולים ומחלקות** 26](#_Toc162339560)

[**מודולים ומחלקות מיובאים** 26](#_Toc162339561)

[**מודולים ומחלקות שפותחו בפרויקט** 27](#_Toc162339562)

[**4.2.** **קטעי קוד עבור אלגוריתמים מרכזיים** 28](#_Toc162339568)

[**4.3.** **בדיקות** 30](#_Toc162339575)

[**5.** **מדריך למשתמש** 30](#_Toc162339576)

[**5.1.** **עץ קבצים** 30](#_Toc162339582)

[**5.2.** **תרשים מסכים** 31](#_Toc162339583)

[**5.3.** **צילום מסך פתיחה והסבר GUI** 31](#_Toc162339591)

[**5.4.** **צילום מסכי האפליקציה בליווי הסברים על פקדים** 31](#_Toc162339592)

[**5.5.** **הבנת flow עבודה במערכת** 31](#_Toc162339593)

[**5.6.** **התקנת המערכת** 31](#_Toc162339604)

[**5.7.** **משתמשי המערכת** 31](#_Toc162339616)

[**6.** **סיכום אישי/רפלקציה** 31](#_Toc162339617)

[**7.** **ביבליוגרפיה** 32](#_Toc162339618)

[**8.** **נספחים** 32](#_Toc162339619)

# **מבוא**

## **ייזום**

### **תקציר הפרויקט**

הפרויקט הוא שירות ענן באינטרנט המאפשר ללקוח לגשת ולהשתמש במשאבי מחשב משותפים ולאחסן מידע על רשת האינטרנט. הפרויקט מיועד לתמוך בשימוש מרובה לקוחות ומתאים למגוון תרחישים בהם שיתוף פעולה וגמישות הם חיוניים. הגישה למרחב הפרויקט תתבצע על ידי רישום של לקוח למערכת עם שם משתמש וסיסמא. הפרויקט מבטיח שמירה מאובטחת על נתוני הלקוח בשימוש בהצפנה מקצה לקצה עבור כל המידע שזורם במערכת. כל הקבצים שיעלה הלקוח ישמרו באזורים מוקצים בהתאמה על גבי ה-DataBase. כל זאת יוצג למשתמש בממשק גרפי בשימוש ב-Tkinter. כמו, כן באמצעות יצירת קבוצות ניתן לעבוד עם כמה משתמשים תחת סביבת עבודה שיתופית.

המוצר הסופי יהיה פלטפורמה המאפשרת לשתף קבצים מכל הסוגים בין לקוחות שונים וסביבת אחסון אישית על גבי הענן באופן מאובטח.

בחרתי בפרויקט זה מתוך רצון להעמיק בנושא הענן וכדי לאתגר את עצמי.

משמעות השם CloudAV היא Cloud שזה ענן באנגלית ועב שזה ענן בעברית.

### **הגדרת הלקוח**

לקוח בפרויקט הוא משתמש פרטי או עסקי אשר מחפש פתרון ענן איכותי ומאובטח לאחסון ושיתוף קבצים. הוא מחפש פלטפורמה שתספק לו את היכולת לשמור ולשתף קבצים בין מספר משתמשים, תוך הבטחת אבטחת מידע גבוהה. הלקוחות יכולים להיות קבוצות עבודה, חברות, גופים עסקיים קטנים וגדולים, שמחפשים פתרון לאחסון ושיתוף קבצים בצורה מאובטחת וגמישה. הפרויקט מתאים לאנשים שמעוניינים לשמור על המידע שלהם באופן מאובטח ולהיות מסוגלים לגשת אליו מכל מקום בעולם באמצעות האינטרנט.

### **יעדים ומטרות**

**היעד המרכזי של הפרויקט שלי הוא בניית מוצר המאפשר שיתוף כל סוגי הקבצים בין לקוחות על גבי הענן באופן פשוט ומאובטח.**

**רישום/ כניסת משתמשים למערכת:** המערכת תאפשר לכל לקוח להתחבר לחשבון שלו עם שם משתמש וסיסמה כדי שיוכל לגשת לאזור האישי שלו, ובכלליות על מנת להיות מזוהה עם המערכת. כל המידע הרגיש הזה יוצפן.

**יצירת קבוצה לשיתוף קבצים:** קבוצת משתמשים יוכלו ליצור סביבת עבודה משותפת בה יוכלו לשתף קבצים על גבי רשת האינטרנט אל השרת. סביבת העבודה תכלול בתוכה כלים אשר ייעלו את העבודה כגון צ'אט, רשימת מטלות, Activity log.

כמו כן, כל הקבצים יעברו הצפנה על מנת לשמור על פרטיות המשתמשים.

**ניהול/ עריכה קבצים של משתמשים אחרים בשרת:** המשתמש יוכל לבצע פעולות שונות על הקבצים בענן. פעולות כמו העברת קובץ ממקום למקום, שינוי שם, ואם מדובר במנהל הקבוצה יוכל גם לשנות הרשאות עבור גישה לקובץ.עריכת הקבצים תתאפשר ע"י עורך שיסופק למשתמשים.

**יצירת תקשורת בין לקוחות:** בשימוש בקבוצות הלקוחות יוכלו לתקשר זה עם זה באמצעות קבצים וצ'אט.

**שימוש אישי:** עבור כל לקוח רשום יוקצה אזור אישי ובו יוכל הלקוח לנהל את קבציו באופן אישי, תוך כדי שימוש בכל הממשקים המוצעים במערכת.

**כל השירותים יועברו דרך ממשק גרפי ידידותי למשתמש**.

### **בעיות תועלות וחסרונות**

אחת מהבעיות המרכזיות שהמערכת נותנת מענה עבורו הוא תקשורת קבצים בין עובדים. ללא סביבת ענן, כל עובד יצטרך להעביר באופן ידני (כונן קשיח, מייל) את הקבצים לאדם הספציפי שירצו לתקשר איתו. מה אם כמה אנשים צריכים גישה לפרויקט? אופציה זו לא תהיה אפשרית ללא שימוש בענן. הענן הופך את שיתוף הקבצים לפשוט ומאפשר קבלה של כמה משתמשים לאותו מידע בפעולה אחת.

המטרה המרכזית של האפליקציה היא באמת להפוך את חווית שיתוף הקבצים לחוויה פשוטה, יעילה והכי חשוב מאובטחת.

מטרות המערכת יתממשו ע"י השירותים הבאים: עריכת קובץ מכמה לקוחות בו-זמנית, סנכרון שותף של הקבצים בענן ותקשורת מהירה ונוחה בין המשתמשים הם חלק מהבעיות המגיעות עם הפרויקט הזה. באופן כללי המטרה היא להקל על הלקוח כמה שיותר ולאפשר תקשורת חלקה וחווית משתמש נוחה.

כל המידע החשוב שרץ בין השרת ללקוח יעבור הצפנה ויבטיח אבטחה מקצה לקצה.

### **פתרונות קיימים**

קיימים פתרונות רבים כמו: [Google Drive](https://www.google.com/drive/), [OneDrive](https://www.microsoft.com/en-us/microsoft-365/onedrive/online-cloud-storage), [Dropbox](https://www.dropbox.com/official-teams-page?_tk=paid_sem_goog_biz_b&_camp=1027825805&_kw=dropbox|e&_ad=676240574557||c&gad_source=1&gclid=EAIaIQobChMIu_y-jsLfhAMVX5KDBx2M3QdyEAAYASAAEgJmaPD_BwE) ועוד. הפתרונות הקיימים ברשת עוצבו עבור חברות מרובות משתמשים ומאפשרים עבורם סביבת עבודה מקצועית, מהירה ומאובטחת. החברות הגדולות עבודות בשיתוף עם מוצרים שונים כמו office365 ושירותי הקבצים של גוגל כדי להפוך את חווית המשתמש לחוויה מרוכזת באפליקציה אחת.

הפרויקט שלי תוכנן סביב הפתרונות הקיימים ומציע את מגוון הפיצ'רים ששאר הפתרונות מציעים (עבודה בקבוצות, שליחת קבצים וכו').

### **טכנולוגיית הפרויקט**

מערכות ענן ושמירת קבצים על גבי האינטרנט קיימות כבר שנים רבות. אופן ההגדרה של המערכת שלי תואם את הצרכים עליהם המערכת שלי עונה, כמובן שכל מערכת מוגדרת באופן שונה

**טכנולוגיית הפרויקט שלי מתעסקת בתחומים רבים כגון:**

ממשק גרפי Tkinter – הממשק הגרפי איתו בחרתי לעבוד במערכת.

בסיס נתונים ב-Sqlite3 – השפה בה נכתב בסיס הנתונים

מודול socket – העברת מידע בין השרת ללקוח ובין הלקוח לשרת בפרוטוקול .TCP/IP

### **תחומי הפרויקט**

המערכת נותנת מענה עבור: עבודה של מספר משתמשים, עריכת קבצים, עבודה בענן, עבודה עם קבצים מכל הסוגים, תקשורת על גבי האינטרנט תוך שימוש במגוון מהפרוטוקולים שלו, אך לא תיתן מענה עבור: תקשורת קולית בין משתמשים.

* ארכיטקטורת שרת-לקוח לאפליקציה מאובטחת.
* ממשק גרפי Tkinter כמסך תצוגה ללקוח.
* בסיס נתונים ב-Sqlite.
* תקשורת בין מחשבים באמצעות שרת ובסיס נתונים ברשת הציבורית.
* פרוטוקול IP/TCP לתקשורת מאובטחת.
* הצפנת מידע לאבטחתו.
* שימוש בפרוטוקול SMTP לשליחת אימיילים ללקוחות.
* אימות דו-שלבי.
* הצפנת סיסמאות באמצעות Hash.
* שימוש ב-threading לניהול תהליכים מרובים.
* שמירת קבצים וקריאתם בצורה מאובטחת.



## **אפיון המערכת**

### **תיאור הפרויקט**

המערכת תיצור עבור הלקוח סביבת עבודה עם שרת ענן באופן חלק ומהיר. כדי לגשת לשירותי המערכת הלקוח ירשם תחילה למערכת באמצעות שם משתמש וסיסמה אשר יועברו בהצפנת Hash ויאוחסנו ב-.DataBase אם יבחר, הלקוח יוכל להפעיל אימות דו-שלבי :הלקוח יכניס את כתובת ה-Email שלו ובשימוש בפרוטוקול SMTP ישלח אל הלקוח קוד באורך 6 תווים אותו יצטרך לספק למערכת כדי להיכנס. ללקוח מוקצה שטח אישי מוגבל ב-Database בו יוכל להעלות בעזרת להוסיף פרוטוקול כל סוג של קובץ שיבחר/ הורדה למחשב הלקוח כל קובץ שהעלה אל שטחו בענן. הלקוח יוכל לארגן את הקבצים שהעלה ע"י יצירת תיקיות. המערכת נותנת מענה לשימוש של מספר לקוחות יחד. לקוח יוכל ליצור קבוצה עם לקוחות אחרים שיבחר. לקבוצה יוקצה שטח אחסון משותף על גבי ה-DataBase בו יוכלו להעלות ולהוריד כל קובץ שיבחרו. המערכת תאפשר עבודה במקביל בשימוש ב-threads כך שמספר משתמשים יכולים לעבוד על אותו קובץ במקביל. בעבודה בקבוצה יוכלו הלקוחות לתקשר זה עם זה ע"י שימוש בהודעות איך. הלקוחות יוכלו לשוחח עם כל חברי הקבוצה ובאופן אישי עם לקוחות אחרים. ההודעות יועברו הצפנה מקצה לקצה באמצעות איך. יוצר הקבוצה יוכל להוסיף/ להסיר לקוחות אחרים בכל זמן שיבחר וגם להגביל גישות לקבצים/תיקיות מסוימות עבור לקוחות שיבחר. כל זאת יוצג למשתמש בממשק גרפי ידידותי ונוח בעל חלונות שונים עבור פעולות שונות בשימוש ב-Tkinter. בתור הרחבה אשתמש בטכנולוגיית Filesystem in Userspace - ממשק תוכנה המאפשר למשתמשים שאינם יכולים לקבל הרשאות מערכת ליצירת מערכות קבצים באופן עצמאי. בעזרת הממשק ייווצר לכל משתמש תיקייה במחשב אליה יוכל לגרור קבצים הישר משולחן העבודה שלו ותהליך השמירה עם השרת יתבצע ברקע.

### **מאפיינים ויכולות**

* הלקוח יוכל ליצור חשבון מכל מחשב שיבחר לעבוד עליו.
* הלקוח יוכל להתחבר לחשבון מכל מחשב שיבחר לעבוד עליו
* הלקוח יוכל להעלות את כל סוגי הקבצים אל האיזור האישי שלו.
* הלקוח יוכל להעלות את כל סוגי הקבצים אל קבוצות משותפות בהן הוא נמצא ולסדר אותם כיצד שיבחר בתיקיות.
* הלקוח יוכל להוריד את כל הקבצים ששמר במערכת אל המחשב האישי שלו
* הלקוח יוכל להוריד את כל הקבצים ששותפו איתו בקבוצות אל המחשב האישי שלו
* השרת יתמוך בהורדה של מספר משתמשים במקביל
* השרת יתמוך בהעלה של מספר משתמשים במקביל
* בעבודה בקבוצה הלקוח יוכל לתקשר עם חברי הקבוצה בהודעות
* לכל קבוצה יהיה admin שיוכל לנהל את הקבוצה עם יכולות מיוחדות של:

הוספה\ הסרה של משתמשים מהקבוצה

עריכת שם הקבוצה

הגדרת הגבלת שימוש של קבצים עבור משתמשים מסוימים

* הלקוח יוכל לעצב בהתאם לנוחותו את מרחב העבודה האישי שלו באופן התצוגה, צבעי נושא, שומרי מסך וכו'.
* לקוח יוכל להעלות קבצים לענן ע"י שימוש בתקיית האחסון שתיווצר עבורו במחשב האישי שלו (.(FUSE

### **בדיקות המערכת**

### **תכנון הבדיקה:**

* קביעת מטרות הבדיקה והפרמטרים לכל סוג של בדיקה.
* יצירת סט תרחישי בדיקה המכסה את כל פונקציות המערכת.
* חלוקת הבדיקות ליחידות קטנות ומוגדרות.

**הכנת הסביבה:**

* הקמת סביבת בדיקה נפרדת או השתמשות בסביבה ניתנת להפעלה והתקנת מערכת המבוססת על הפרויקט.
* התקנת כלי בדיקה, סימולטורים, ותוכנות ניהול בדיקות.

**ביצוע הבדיקות:**

* בדיקות ידניות: ביצוע תרחישי בדיקה ידניים לבדיקת פונקציונליות, נגישות ואבטחה.
* בדיקות אוטומטיות: הרצת סקריפטים אוטומטיים המבצעים תרחישי בדיקה מקיפים על פי תכנית הבדיקה.
* מעקב אחר תוצאות הבדיקה ותיעודן במערכת ניהול בדיקות.

**אנליזת תוצאות:**

* בדיקה והשוואה של תוצאות הבדיקה לציפיות המצוינות במטרות הבדיקה.
* זיהוי ותיקון שגיאות ותקלות.

### **תכנון וניהול לוח זמנים**

|  |  |
| --- | --- |
| תכנונים: | לוח זמנים: |
| יצירת שרת פשוט שידע לקבל/לשלוח קבצים עם לקוח (הקובץ ישמר על המחשב עצמו עד הקמת ה-DataBase). | **1/11/23** |
| התאמת המערכת בשימוש של threads לעבודה במקביל עם כמה לקוחות. | **1/11/23** |
| הגדרת DataBase ראשוני עבור לקוח יחיד | **24/11/23** |
| יצירת סביבת עבודה ראשי GUI | **1/12/23** |
| הוספת הרשמה למערכת עבור כל משתמש עם כל הבדיקות הנחוצות + הוספת הצפנה (Hash functions). | **13/12/23** |
| עדכון ה-DataBase לעבודה עם מספר משתמשים (הקצאת שטח עבור כל משתמש + הוספת טבלאות של שמות משתמשים וסיסמאות). | **17/12/23** |
| יצירת חלון התחברות/ הרשמה GUI | **24/12/23** |
| עדכון ה-DataBase לעבודה עם מספר משתמשים (הקצאת שטח עבור כל משתמש + הוספת טבלאות של שמות משתמשים וסיסמאות. | **1/1/24** |
| שיפור GUI (יצירת חלון התחברות). | **6/1/24** |
| הוספת היכולת בצד השרת ליצור קבוצה של לקוחות בסיסית. | **15/2/24** |
| הוספת היכולת בצד השרת ליצור קבוצה של לקוחות בסיסית. | **20/4/24** |
| הגדרת הרשאות בין יוצר הקבוצה (admin) לשאר הלקוחות (יכולות ה-admin). | **29/4/24** |
| עדכון ה-DataBase לעבודה עם קבוצות (הקצאת שטחים לקבצי הקבוצה ולפרטים של לקוחות הקבוצה). | **1/4/24** |
| שיפור GUI (הוספת חלון קבוצתי עם המאפיינים של | **12/5/24** |

### **ניהול סיכונים**

**גישה למידע פרטי של משתמש ע"י משתמשים אחרים.**  הפתרון לסיכון כרוך בעובדה שרק השרת יכול לתקשר עם ה-Database. בסיום ההתחברות ל- thread של השרת מועבר המזהה האישי של המשתמש, כך שלא קורה מצב של הצטלבות בין משתמשים שתוביל לפגיעה בפרטיות המידע של המשתמש. כמו כן, המידע יוצפן בהמשך. בנוסף, שמירה על הפרטיות באה בהצפנת הסיסמאות באמצעות הצפנה (SHA-256).

**העמסה יזומה על המערכת.** הפתרון: בחלון ההתחברות המשתמש יוגבל לכמות ניסיונות מסוימת, כך שהתקפות כמו Brute force, לא יפגעו במערכת. כמו כן, תתקיים הגבלה על גודל הקבצים (הגבלה שתמנע מקרי קיצון ולא תפגע בתפקוד המשתמש).

**ערבוב בין קבצים של משתמשים שונים.** הפתרון: עבור כל משתמש נוצר Table יחודי ב-Database. הטבלאות הייחודיות לכל משתמש יצרו הפרדה בין קבצי כל משתמש, ובכך ימנע ערבוב בין הקבצים השונים של המשתמשים

**שליחה וקבלה חלקית של קבצים \ אישור קבלת המידע.** ע"י שימוש בדגלים בפרוטוקולי השליחה, הקבלה, ההתחברות וההרשמה השרת יוכל להודיע ללקוח על מצב הפעולה שהוא מבצע ובכך יציבות הפרויקט ורמת הסיכון לאיבוד מידע יורדת.

**התמודדות עם שגיאות.** לאורך כל הקוד יש שימוש ב-try/except/finally, וכך השרת יודע איך להתמודד עם שגיאות של הלקוח, להודיע ללקוח על השגיאה, ולהוביל אותו לפתרון אחר.

# **תיאור מפורט של המערכת**



## **פירוט וניתוח יכולות המערכת**

**פירוט היכולות בצד שרת**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם היכולת** | **מהות היכולת** | **פעולות הנדרשות למימוש היכולת** | **אובייקטים נחוצים** |
| הוספת מידע ל-Database. | להוסיף מידע ל-Database. | קליטת נתונים, פענוח, הצפנה. | פענוח, הצפנה, תקשורת, Database. |
| שליפת מידע מה-Database. | לשלוף מידע מה-Database. | קליטת נתונים, פענוח, הצפנה, שליחה ללקוח. | פענוח, הצפנה, תקשורת, Database. |
| שינוי מידע ב-Database. | לשנות מידע מסוים ב- Database. | קליטת נתונים, פענוח, הצפנה. | פענוח, הצפנה, תקשורת, Database. |
| הרשמה | רישום המשתמש  ב-Database. | קליטת נתונים, פענוח, בדיקת כפילויות, הוספת מידע ל-Database, הצפנה, שליחת תשובה ללקוח. | פענוח, הצפנה, תקשורת, Database. |
| התחברות | חיבור הלקוח לחשבון המתאים לו. | קליטת נתונים, פענוח, בדיקת התאמה, שליפת מידע מה-Database, הצפנה, שליחת תשובה ללקוח. | פענוח, הצפנה, תקשורת, Database. |
| אימות דו-שלבי | שליחת קוד אימות ל-Email של הלקוח. | קליטת נתונים, פענוח, שליחת מייל, בדיקת תאימות, שליחת תשובה ללקוח. | פענוח, הצפנה, תקשורת, |
| שליחת קבצים ללקוח | שליחת הקובץ המבוקש אל הלקוח | קליטת נתונים, פענוח, בדיקת תקינות, שליפת מידע מה-Database, שליחת תשובה ללקוח. | פענוח, הצפנה, תקשורת, Database. |
| קבלת קבצים מהלקוח | קבלת קבצים שהלקוח שלח לשרת | קליטת נתונים, פענוח, שמירה ב-Database, תשובה למשתמש | פענוח, הצפנה, תקשורת, Database. |
| מחיקת קובץ | למחוק את הקובץ המבוקש | קליטת נתונים, פענוח, בדיקת תקינות, שינוי מידע ב-Database, שליחת תשובה ללקוח. | פענוח, הצפנה, תקשורת, Database. |

**פירוט היכולות בצד לקוח**

|  |  |  |  |
| --- | --- | --- | --- |
| **שם היכולת** | **מהות היכולת** | **פעולות הנדרשות למימוש היכולת** | **אובייקטים נחוצים** |
| הרשמה | רישום המשתמש  ב-Database. | ממשק הרשמה גרפי, בדיקת תקינות, הצפנה, שליחה לשרת, פענוח, קבלת תשובה מהשרת. | ממשק הרשמה גרפי, פענוח, הצפנה, תקשורת. |
| התחברות | חיבור הלקוח לחשבון המתאים לו. | ממשק התחברות גרפי, בדיקת תקינות, הצפנה, שליחה לשרת, פענוח, קבלת תשובה מהשרת. | ממשק התחברות גרפי, פענוח, הצפנה, תקשורת. |
| שליחת קבצים לשרת | שליחת הקובץ המבוקש אל השרת | ממשק שליחת קבצים גרפי, בדיקת תקינות, הצפנה, שליחה לשרת, פענוח, קבלת תשובה מהשרת. | ממשק שליחת קבצים גרפי, פענוח, הצפנה, תקשורת. |
| קבלת קבצים מהשרת | קבלת קבצים שהשרת שלח ללקוח | ממשק שליחת קבצים גרפי, בדיקת תקינות, הצפנה, שליחה לשרת, פענוח, קבלת תשובה מהשרת. | ממשק קבלת קבצים גרפי, פענוח, הצפנה, תקשורת. |
| מחיקת קובץ | למחוק את הקובץ המבוקש | ממשק שליחת קבצים גרפי, בדיקת תקינות, הצפנה, שליחה לשרת, פענוח, קבלת תשובה מהשרת. | ממשק מחיקת קבצים גרפי, פענוח, הצפנה, תקשורת. |

# **ארכיטקטורה**

פירוט ההחלטות שנלקחו למימוש המערכת (עיצוב:(



## **ארכיטקטורת המערכת**

**תיאור החומרה**

* המידע זורם מהמערכת של הלקוח לשרת.
* המידע זורם מהשרת למערכת של הלקוח.
* המידע זורם מהשרת לבסיס הנתונים.
* המידע זורם מבסיס הנתונים לשרת.

**סכימת בלוקים**  
תמונה שמכילה תרשים, שרטוט, שרטוט טכני, תוכנית

התיאור נוצר באופן אוטומטי

## **טכנולוגיית המערכת**

**שפת תכנות**

שפת התכנות שעל גביה בחרתי ליצור את הפרויקט היא python.

עבור ה-Database השתמשתי בשפת Sqlite.

**מערכת הפעלה**

מערכת ההפעלה שעליה רץ הפרויקט היא Windows. סיבת הבחירה ב-Windows נבעה מכך שהפרויקט נבנה ונוסה על מחשבי Windows, לכן הרצת התוכנית במערכות אחרות כמו Linux או MacOS לא תיהיה וודאית.  
**תקשורת**

התקשורת בין השרת ללקוח מתבצעת ע"י פרוטוקול תקשורת בשכבת התעבורה – TCP/IP. בשימוש באלגוריתם שליחה וקבלה שיצרתי

## **זרימת המידע במערכת**

* 1. המידע זורם מהלקוח לשרת
  2. המידע זורם מהשרת אל ה-database
  3. המידע זורם מה-database לשרת
  4. המידע זורם מהשרת ללקוח

תמונה שמכילה תרשים, שרטוט, שרטוט טכני, תוכנית

התיאור נוצר באופן אוטומטי



## **אלגוריתמים עיקריים**

בעיה אלגוריתמית שנתקלתי בה בפרויקט הייתה כיצד לוודא שכל קובץ עובר בשלמותו מהשרת אל הלקוח באופן מסודר ויעיל. בפרוטוקול TCP המידע נשלח כ-stream ארוך של ביטים, לכן סדר בגודל הBuffer- ומיון נכון של המידע נחוש כדי להצליח.

פתרון אפשרי היה שבכל בקשה של שליחה/קבלת קובץ הצד השולח היה מוסר מאפיינים חשובים כגון: גודל הקובץ, מספר הקבצים שנשלחים וסיומת הקובץ לנמען, ובכך הנמען יוכל להתאים את האלגוריתם לצרכיו האישיים כך שהמידע יזרום באופן תקין.

בסופו של דבר בחרתי באלגוריתם מהיותו חשוף פחות לפגיעות ולשגיאות במהלך הריצה.

אלגוריתם ההצפנה איתו בחרתי לעבוד בהצפנת סיסמאות המשתמשים הוא

SHA-256. הצפנת Hash היא חד כיוונית – המידע שמוצפן לא יכול לחזור לצורתו המקורית לאחר ההצפנה. בכך אוכל לשמור על פרטיות המשתמשים מפני כל אדם, אפילו מפני מפתחי הפרויקט.



## **סביבת הפיתוח**

**כלי פיתוח נדרשים**

דרוש מהמשתמש מחשב בעל מערכת הפעלה Windows, חיבור לאינטרנט (עבור הפרויקט שלי גם חיבור LAN מקומי מספיק), תוכנת Python, מותקנת על המחשב וחשבון Email.

**סביבת הבדיקות**

בסביבת הבדיקה נרשים לפחות 2 מחשבי Windows שמחוברים לרשת אינטרנט מקומי אחידה, בעלי Python מותקן.

## **פרוטוקול התקשורת**

**פרוטוקול TCP/IP :**

מידע שנשלח מהלקוח לשרת, או מהשרת ללקוח כמו פרטי משתמש, פרטי קבצים, קבצי BLOB המגיעים ממסד הנתונים, הודעות של משתמשים וכו' נשלח ע"י הפרוטוקול. השימוש בפרוטוקול נובע מהעובדה שסדר המידע מגיע לשרת כפי שנשלח מהלקוח.

**פרוטוקול SMTP:**

עבור Two Factor Authentication אני אשתמש בפרוטוקול SMTP כדי לשלוח למייל המשתמש הודעת אישור כניסה לחשבון המשתמש. הפרוטוקול ישומש עבור אימות דו-שלבי עם המשתמש. באמצעות הפרוטוקול התקשורת בין הEmail- של הלקוח והשרת מתבצעת. ללקוח ישלח מייל בעל קוד אותו יצטרך להקיש כדי להיכנס לחשבונו

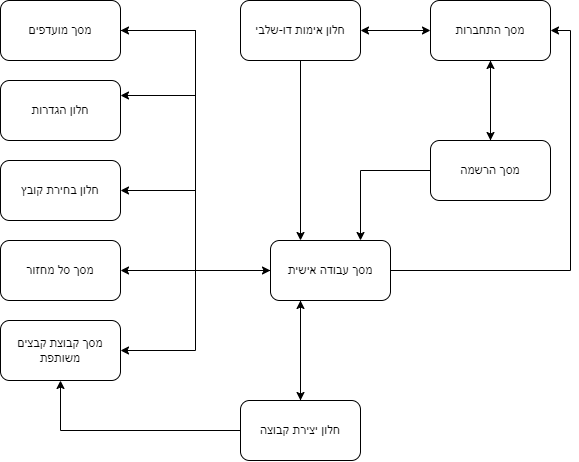
**פרוטוקול אישי שלי:**

עקב הסבירות שמידע שנשלח ב-socket TCP יכול להתחבר עם מידע אחר, מה שיגרום לבלבול בסדר המידע, הכנתי פרוטוקול אשר פותר את הבעיה הזו

בקצרה, הפרוטוקול משתמש בשתי חלקים לשליחת מידע: תחילה, נשלחת אורך המידע ככותרת 4 בתים (המייצגת מספר שלם בסדר סופי), ואחריה נשלחת המידע המסודר. בקצה המקבל, תחילה מתקבלת הכותרת של האורך כדי לקבוע את גודל המידע המצוי בדרך, ואז מתקבל המידע עצמו ומתבנה מחדש. השימוש ב-`pickle.dumps` מאפשר לפרוטוקול להתמודד עם נתונים מסוג string ו־bytes.



## **מסכי המערכת/ממשק משתמש**



**מסך התחברות:**

תמונה שמכילה צילום מסך, מחשב, אדם

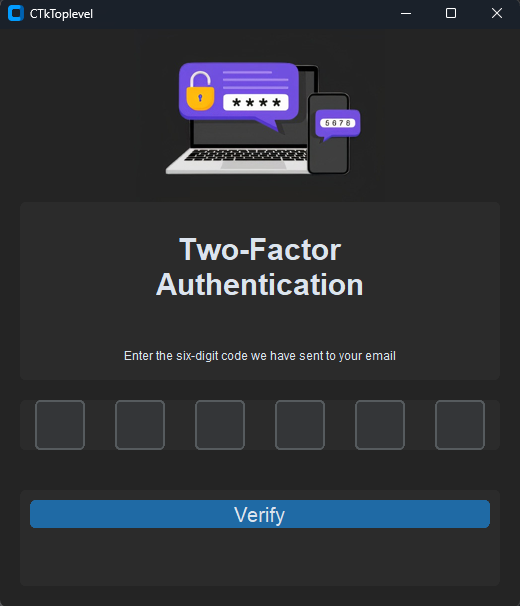
התיאור נוצר באופן אוטומטיתיאור המסך: המסך הראשון שהלקוח רואה הוא מסך ההתחברות, המערכת מניחה כי הלקוח התחבר בעבר למערכת ולכן יש לו משתמש רשום, אך למקרה והלקוח הוא לקוח חדש ניתן לעבור למסך הרשמה. במסך הזה המשתמש מכניס את הפרטיים האישיים המבוקשים כדי להתחבר למשתמש שלו. במידה והמידע שהמשתמש הזין איננו נכון, תופיע הודעת שגיאה + פירוט השגיאה.

**מסך הרשמה:**

תמונה שמכילה צילום מסך, מחשב, טקסט, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיתיאור המסך: במסך המשתמש יכניס את הפרטים המבוקשים (Email וסיסמה). במקרה של שגיאות תחביריות תחת כל שדה תופיע הודעת בקרה ובה יהיה כתוב שגיאות בפרטים שהזין המשתמש (אם יש כאלה).

**חלון אימות דו-שלבי:**

החלון מכיל מקום למילוי קוד באורך 6 תווים שישלח למשתמש במייל. מתחת לכפתור האימות תופיע שגיאה במקרה והקוד איננו נכון

**מסך ראשי:**

תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטיתיאור המסך: במסך הראשי המשתמש יראה את כל הקבצים שהעלה בעבר אל הענן ששמורים ב-Database (אם יש כאלה) ברשימה מסודרת עם פרטי הקבצים. בלחיצה על קובץ תיפתח רשימה של פעולות שניתן לעשות על הקובץ (מחיקה, הורדה, שיתוף). בצד המסך תיהיה לשונית ארגון של מבנה התיקיות של הלקוח בשטח האחסון שלו ולשונית הקבוצות.

**מסך יצירת קבוצה:**

במסך יהיה שדה בו המשתמש מזין את השם של הקבוצה שיבחר. מתחתיו יש רשימה של כל המיילים של המשתמשים הקיימים במערכת ומביניהם היוצר של הקבוצה יבחר את מי ירצה להוסיף

**תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי**



## **מסד הנתונים**

**טבלת Aauthenticated:**

**תיאור:** טבלה זו מאחסנת פרטי אימות משתמש עבור מערכת.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם עמודה | סוג נתונים | תיאור | חובה | דוגמה |
| id | INTEGER | מזהה ייחודי עבור רשומת המשתמש (מפתח ראשי). | כן | 1 |
| דוא״ל (email) | TEXT | כתובת הדוא״ל של המשתמש. | כן | ] כתובת הדוא״ל מוסרת] |
| שם משתמש (username) | TEXT | שם המשתמש שנבחר על ידי המשתמש לכניסה. | כן | johndoe |
| סיסמה (password) | TEXT | דיבוב SHA-256 של הסיסמה המקורית. לעולם לא תאוחסן הסיסמה עצמה. | כן | (ערך מוצפן) |

**טבלת Files:**

**תיאור:** טבלה זו מאחסנת מידע אודות קבצים במערכת.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם עמודה | סוג נתונים | תיאור | חובה | דוגמה |
| id | INTEGER | מזהה ייחודי עבור רשומת הקובץ (מפתח ראשי). | כן | 123 |
| בעלים (Owner) | TEXT | בעל הקובץ. | כן | jane.doe |
| שם (Name) | TEXT | שם הקובץ. | כן | my\_report.pdf |
| גודל (Size) | INTEGER | גודל הקובץ בבתים. | כן | 1048576 (1MB) |
| תאריך (Date) | TEXT | התאריך שבו הקובץ הועלה. | כן | 2024-03-10 |
| מועדף (Favorite) | INTEGER | דגל המציין אם הקובץ מועדף (0 - לא מועדף, 1 - מועדף). | לא (ברירת מחדל 0) | 1 |
| תכולת קובץ (FileBytes) | BLOB | נתונים בינאריים של תכולת הקובץ. | לא | (לא מוצג, נתונים בינאריים) |

**טבלת GroupFiles:**

**תיאור:** טבלה זו מאחסנת מידע על אודות קבצים המשויכים לקבוצות ספציפיות במערכת.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם עמודה | סוג נתונים | תיאור | חובה | דוגמה |
| id | INTEGER | מזהה ייחודי עבור רשומת הקובץ (מפתח ראשי). | כן | 456 |
| בעלים (Owner) | TEXT | בעל הקובץ. | כן | jsmith |
| שם (Name) | TEXT | שם הקובץ. | כן | team\_plan.docx |
| גודל (Size) | INTEGER | גודל הקובץ בבתים. | כן | 524288 (512KB) |
| תאריך (Date) | TEXT | התאריך שבו הקובץ הועלה. | כן | 2024-03-08 |
| שם קבוצה (GroupName) | TEXT | שם הקבוצה שאליה שייך הקובץ. | כן | Marketing Team |

**טבלת Rooms:**

**תיאור:** טבלה זו מאחסנת מידע על אודות חדרים במערכת.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| שם עמודה | סוג נתונים | תיאור | חובה | דוגמה |
| Id | INTEGER | מזהה ייחודי עבור רשומת הקובץ (מפתח ראשי). | כן | 796 |
| Admin | TEXT | שם משתמש או ID של מנהל הקבוצה | כן | Klee |
| Name | TEXT | שם הקבוצה | כן | Training Room |
| Participants | TEXT | רשימה של חברי הקבוצה (יכולה להיות ריקה) | כן | alice.wong, bob.lee |
| Permissions | TEXT | רשימה של הרשאות עבור חברי הקבוצה (יכולה להיות ריקה) | כן | [1,1,0,1] |



## **חולשות ואיומים**

**שכבת האפליקציה:**

הסיסמאות עוברות הצפנת hash ורק לאחר מכן נשלחים לשרת ומשם לdatabase , ומכיוון שלא ניתן לשחזר סיסמאת HASH הסיסמא בטוחה

**שכבת התעבורה:**

מכיוון ששאר הדברים כרגע לא עוברים הצפנה המערכת אכן חשופה לדליפות של מידע במקרים של man in the middle', אך בהמשך המידע יוצפן. הצפנת המידע תועיל בכך שגם אם יהיה אדם באמצע שיחבל בתקשורת, המידע שזורם יהיה מוצפן.

השרת והלקוח עושים בינהם three way handshake לפני פעילות כך שהתקשורת עוברת בדיקה לפני פעולה

במקרה של התקפת DDOS המערכת תחווה עומס שיפגע בתפקודה ויכול להוביל לקריסת השרת. עבור מקרים כאלה המערכת שלי איננה מוגנת

# **מימוש הפרויקט**

פירוט מימוש המערכת (המשך שלב העיצוב)



## **מודולים ומחלקות**

**מודולים ומחלקות מיובאים**

* Pickle: שומר/טוען נתוני פיתון.
* Socket: תקשורת רשת.
* :Struct אורז/פורק נתונים בינאריים.
* Threading: מפעיל משימות מרובות במקביל.
* :Queue תור משימות עבור תהליכונים.
* :Filedialog תיבות דו-שיח לקבצים עבור אינטראקציית משתמש.
* Datetime: טיפול בתאריך ושעה.
* Customtkinter: משפר ווידג'טים של Tkinter (ממשק).
* Scrolled: מסגרת לגלילה עבור תוכן.
* Re: ביטויים רגולריים לתבניות טקסט.
* Ttkbootstrap: הרחבה ל-Tkinter
* PIL: ספריית תמונות פיתון (לתמונות).
* Tkinter: ערכת כלים סטנדרטית של פיתון לממשקי משתמש גרפיים.
* Tkvideo: מנגן סרטונים ביישומי Tkinter.
* sqlite3: מנוע מסד נתונים

**מודולים ומחלקות שפותחו בפרויקט**

**ClientComunication:**

**תפקיד:** לקשר בין השרת לממשק הגרפי.

**יתרונות:**

* ניהול סדור של פקודות שפונות לשרת.
* הפרדה בין הממשק הגרפי ללקוח.
* מניעת קריסות מיותרות.

**תכונות המחלקה:**

* Client\_socket: מקבל את ה-socket של הלקוח שמחובר לשרת. דרכו מתבצעת כל התקשורת.

**פעולות:**

* r\_when\_submit(self, attempt\_type, u\_email, u\_username, u\_password)
* l\_when\_submit(self, attempt\_type, u\_email, u\_password)
* send\_file(self, file\_name, short\_filename, short\_file\_date, file\_bytes)
* receive\_checked\_files(self, select\_file\_names\_lst, save\_path)
* notify\_and\_receive\_files(self)
* delete\_checked\_files(self, select\_file\_names\_lst)
* rename\_files(self, rename\_data)
* favorite\_file(self, file\_name, switch\_value)
* get\_all\_users(self)
* create\_group(self, group\_name, group\_participants, permissions)
* get\_all\_groups(self)

**GroupComunication:**

**תפקיד:** לנהל את בקשות הלקוח מהממשק הגרפי מול השרת בקבוצות.

**תכונות המחלקה:**

* Client\_socket: מקבל את ה-socket של הלקוח שמחובר לשרת. דרכו מתבצעת כל התקשורת.
* On\_broadcast\_callback: המידע שהשרת שולח לכל המשתמשים בקבוצה צריך לעבור עיבוד של הממשק הגרפי. התכונה היא פונקציה שמעבדת את כל הפניות של השרת.
* Receive\_thread: Thread לקבצים שמשודרים מהשרת.
* self.running = False - Flag to control the thread.

**פעולות:**

* get\_all\_users(self)
* create\_group(self, group\_name, group\_participants)
* join\_group(self, group\_name)
* leave\_group(self)
* receive\_broadcasted\_files(self, on\_broadcast\_callback)
* send\_file(self, file\_name, short\_filename, short\_file\_date, file\_bytes)
* delete\_checked\_files(self, select\_file\_names\_lst)
* notify\_and\_receive\_files(self)
* rename\_files(self, rename\_data)
* receive\_checked\_files(self, select\_file\_names\_lst, save\_path)

**MainClient:**

**תכונות:**

* Client\_socket: מקבל את ה-socket של הלקוח שמחובר לשרת. דרכו מתבצעת כל התקשורת.
* Client\_comunicator: מחלקת Client Communications.
* Group\_comunicator: מחלקת Group Communications.

**פעולות:**

* Run(self)



## **קטעי קוד עבור אלגוריתמים מרכזיים**

**אלגוריתם Send/Recv:**

**מטרה:**

* לשלוח נתונים באופן אמין דרך חיבור סוקט בפורמט מובנה.

**בעיות שהאלגוריתם פותר:**

* טיפול בסוגי נתונים שונים: תומך בשני נתוני מחרוזת ובתים, ומבטיח שידור תקין.
* הפיכת נתונים לטורי בתים (serialization): ממיר נתוני מחרוזת לטורי בתים באמצעות Pickle לשם עקביות ותאימות.
* קידומת עם אורך: מוסיף כותרת אורך של 4 בתים (בפורמט big-endian) לפני הנתונים, ומאפשר למקבל לדעת במדויק כמה נתונים לצפות.

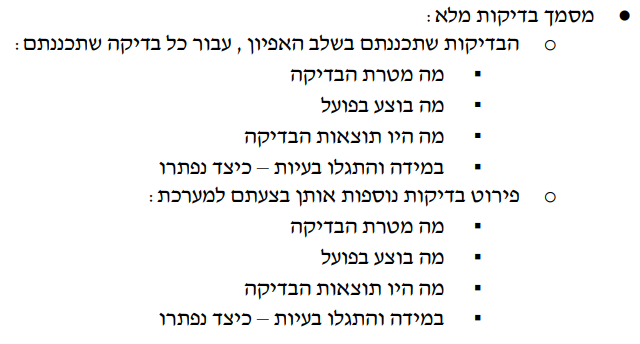
**קוד:**

def send\_data(self, client\_socket: socket, data: str | bytes):  
 if isinstance(data, str):  
 data = pickle.dumps(data)  
  
 data\_len = len(data).to\_bytes(4, byteorder='big')  
 client\_socket.send(data\_len + data)

def recv\_data(self, client\_socket: socket):  
 data\_len = client\_socket.recv(4)  
  
 while len(data\_len) < 4:  
 data\_len += client\_socket.recv(4 - len(data\_len))  
 len\_to\_int = int.from\_bytes(data\_len, byteorder='big')  
 data = client\_socket.recv(len\_to\_int)  
  
 while len(data) < len\_to\_int:  
 data += client\_socket.recv(len\_to\_int - len(data))  
  
 return data



## **בדיקות**



# **מדריך למשתמש**



## **עץ קבצים**

פירוט כלל קבצי המערכת – עץ קבצים

## **תרשים מסכים**

תרשים המתאר את היררכית המסכים



## **צילום מסך פתיחה והסבר GUI**

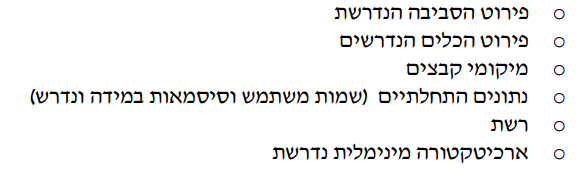
## **צילום מסכי האפליקציה בליווי הסברים על פקדים**

## **הבנת flow עבודה במערכת**

הסבר מפורט המתאר איך המשתמש מפעיל את כל היכולות באפליקציה דרך המסכים (ממשק משתמש)

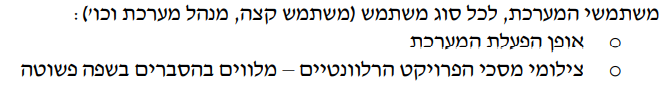


## **התקנת המערכת**

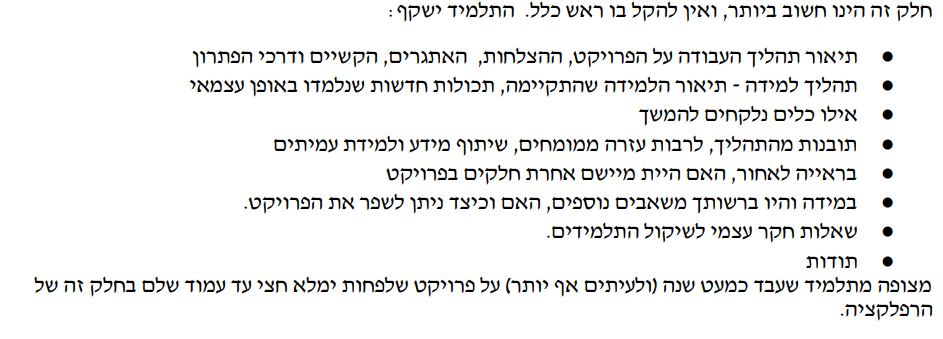




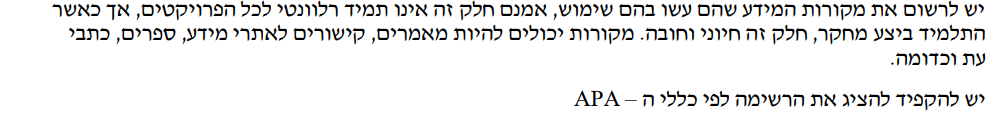
## **משתמשי המערכת**



# **סיכום אישי/רפלקציה**



# **ביבליוגרפיה**



# **נספחים**

