****

**תיק פרויקט – Idan Cloud**

**בית הספר: תיכון הנדסאים הרצליה**

**שם הפרויקט: IdanCloud**

**שם התלמיד: עידן חזאי**

**תעודת זהות: 330905159**

**שם המנחה: אופיר שביט**

**שם החלופה:**

**תאריך הגשה:**

# **תוכן עניינים**

* [תוכן עניינים 2](#_Toc189834938)
* [מבוא 4](#_Toc189834939)

[ייזום: 4](#_Toc189834940)

[תיאור ראשוני של המערכת: 4](#_Toc189834941)

[הגדרת הלקוח: 5](#_Toc189834942)

[יעדים ומטרות: 6](#_Toc189834943)

[בעיות תועלת וחסכנות: 7](#_Toc189834944)

[מערכות ותוכנות קיימות: 7](#_Toc189834945)

[סקירת טכנולוגית: 9](#_Toc189834946)

[תחומים בהם המערכת עוסקת: 10](#_Toc189834947)

[תחומים בהם המערכת לא מטפלת: 11](#_Toc189834948)

[פירוט תיאור המערכת (אפיון): 12](#_Toc189834949)

[תיאור מפורט של המערכת: 12](#_Toc189834950)

[פירוט יכולות לכל משתמש: 16](#_Toc189834951)

[פירוט הבדיקות המתוכננות: 18](#_Toc189834952)

[ניהול ותכנון לו"ז: 22](#_Toc189834953)

[ניהול סיכונים ודרכי התמודדות: 25](#_Toc189834954)

* [תיאור תחום הידע 26](#_Toc189834955)

[פירוט מעמיק של יכולות המערכת: 26](#_Toc189834956)

[יכולות בצד שרת: 26](#_Toc189834957)

[יכולות בצד לקוח: 30](#_Toc189834958)

* [מבנה/ארכיטקטורה של המערכת 35](#_Toc189834959)

[תיאור הארכיטקטורה של המערכת: 35](#_Toc189834960)

[תיאור החומרה: 35](#_Toc189834961)

[שרטוט: 35](#_Toc189834962)

[טכנולוגיה רלוונטית: 36](#_Toc189834963)

[זרימת המידע במערכת: 37](#_Toc189834964)

[תיאור האלגוריתמיקה בפרויקט: 38](#_Toc189834965)

[ניסוח וניתוח של הבעיה האלגוריתמית: 38](#_Toc189834966)

[אלגוריתמים קיימים: 39](#_Toc189834967)

[סקירת הפתרון הנבחר: 40](#_Toc189834968)

[תיאור סביבת הפיתוח: 42](#_Toc189834969)

[כלי הפיתוח הנדרשים: 42](#_Toc189834970)

[הסביבה והכלים הנדרשים לבדיקות: 42](#_Toc189834971)

[תיאור פרוטוקול התקשורת: 43](#_Toc189834972)

[תיאור מילולי של הפרוטוקול: 43](#_Toc189834973)

[פירוט כלל ההודעות הזורמות במערכת: 43](#_Toc189834974)

[תיאור מסכי המערכת: 43](#_Toc189834975)

[תיאור כל מסך: 43](#_Toc189834976)

[תרשים מסכים: 43](#_Toc189834977)

[תיאור מבני הנתונים: 43](#_Toc189834978)

[פירוט מבני הנתונים: 43](#_Toc189834979)

[פירוט מאגרי המידע של המערכת: 43](#_Toc189834980)

[מסד נתונים: 43](#_Toc189834981)

[סקירת חולשות ואיומים: 43](#_Toc189834982)

* [מימוש הפרויקט 43](#_Toc189834983)

[סקירת כל המודולים והמחלקות המרכיבים את המערכת והקשרים ביניהם: 43](#_Toc189834984)

[מודולים מיובאים: 43](#_Toc189834985)

[מודולים שפתחתי: 43](#_Toc189834986)

[קטעי קוד ופיתוחים מיוחדים: 43](#_Toc189834987)

[מסמך בדיקות מלא: 43](#_Toc189834988)

[בדיקות שתוכננו בשלב האפיון: 43](#_Toc189834989)

[בדיקות נוספות: 43](#_Toc189834990)

* [מדריך למשתמש 43](#_Toc189834991)

[קבצי המערכת: 43](#_Toc189834992)

[התקנת המערכת: 43](#_Toc189834993)

[הסביבה הנדרשת: 43](#_Toc189834994)

[הכלים הנדרשים: 43](#_Toc189834995)

[מיקומי קבצים: 43](#_Toc189834996)

[נתונים התחלתיים: 43](#_Toc189834997)

[רשת: 44](#_Toc189834998)

[ארכיטקטורה נדרשת: 44](#_Toc189834999)

[משתמשי המערכת: 44](#_Toc189835000)

[אופן הפעלת המשתמש: 44](#_Toc189835001)

[צילומי מסכי הפרויקט: 44](#_Toc189835002)

* [רפלקציה 44](#_Toc189835003)

[תהליך העבודה: 44](#_Toc189835004)

[תהליך הלמידה: 44](#_Toc189835005)

[כלים נלקחים להמשך: 44](#_Toc189835006)

[תובנות מהתהליך: 44](#_Toc189835007)

[ראייה לאחור: 44](#_Toc189835008)

[איך הייתי משפר: 44](#_Toc189835009)

[שאלות חקר עצמי: 44](#_Toc189835010)

[תודות: 44](#_Toc189835011)

* [ביבליוגרפיה 44](#_Toc189835012)
* [נספחים 44](#_Toc189835013)

# **מבוא**

## **ייזום:**

### **תיאור ראשוני של המערכת:**

הפרויקט שאני בחרתי לעשות הוא מערכת אחסון ענן, בדומה לGoogle Drive, Dropbox ו-OneDrive. מטרת הפרויקט היא לתת לכל אדם המעוניין להתחבר ולשמור קבצים בשרת, בצורה מהירה, נגישה ונוחה לשימוש.

המוצר המוגמר צריך לבצע קודם כל את הבסיס – לתת למשתמש להעלות קבצים ולשמור אותם, וכן למחוק אותם לפי צורכו. בנוסף לבסיס, למוצר יהיה ממשק משתמש נוח לשימוש ולהבנה על ידי אנשים רבים, QOL פיצ'רים לנוחות המשתמש ועוד המון טכנולוגיות מאחורי הקלעים.

בחרתי לעשות את הפרויקט שלי מכיוון שנתקלתי בהמון שירותי אחסון בענן, אך לכל אחד היו חסרונות משלו שמאוד הפריעו לעבוד ולהשתמש בו. בחלק חסרים יכולות שימושיות כמו שיתוף קבצים, אחרים קשים מאוד לשימוש ורישום, חלק דורשים תשלום אפילו לתוכנית הבסיסית ולכן רציתי לבנות בעצמי תוכנה שיכולה לבצע את כל מה שחשוב לדעתי במקום אחד וגם ניתנת לעריכה ולהוספת אופציות על ידי.

לפני שאני אתחיל לכתוב את הקוד של הפרויקט יש כמה אתגרים שיכולים לצוץ. מערכת שיתוף הקבצים יכולה להיות טריקית מכיוון שכאשר מישהו משתף קובץ ולאחר מכן משנה את השם שלו, איך ניתן לעדכן זאת גם אצל האדם שאיתו שיתפו? עוד אתגר שאני צופה הוא יצירת תקשורת לא סינכרונית בין השרת ללקוח. כלומר, עד עכשיו בפרויקטים שעשיתי התקשורת בין השרת ללקוח הייתה סינכרונית – על כל הודעה שהלקוח היה שולח לשרת הוא היה מקבל תשובה כלשהי. בפרויקט הזה ישנן פעולות שיכולות לקחת הרבה זמן כמו לדוגמה העלאת קובץ מאוד גדול, ולכן אני לא רוצה שבזמן הזה הלקוח יהיה תקוע ותקשורת לא סינכרונית פותרת את הבעיה הזאת. תקשורת כזו היא יותר מסובכת שכן צריך לדעת לנהל את כל ההודעות בצד הלקוח.

בנוסף לאתגרים הטכניים, זהו הפרויקט הגדול ביותר שכתבתי, ולכן ניהול הקוד וחלוקתו לקבצים נפרדים ומחלקות יכול להיות מאתגר מאוד.

### **הגדרת הלקוח:**

המערכת מיועדת לאנשים פרטיים וארגונים קטנים עד בינוניים הזקוקים לפתרון אמין ומאובטח לניהול קבצים דיגיטליים. באופן ספציפי, קהל היעד כולל: אנשי מקצוע שצריכים לאחסן ולגשת למסמכים חשובים בצורה מאובטחת. סטודנטים ומחנכים המנהלים תיקים אקדמיים כגון מטלות, עבודות מחקר ומצגות. משתמשים כלליים שרוצים מקום בטוח ונוח לאחסון קבצים אישיים כמו תמונות, סרטונים ותוכן דיגיטלי אחר.

צוותים הדורשים כלים משותפים כדי לשתף ולנהל מסמכים בצורה מאובטחת. עסקים שזקוקים לפתרון פשוט אך מאובטח לארגון ואחסון נתונים רגישים כמו חוזים ודוחות.

עיצוב המערכת נותן עדיפות לנוחות השימוש עבור משתמשים בודדים תוך שמירה על ביטחון המידע הנדרש לשימוש מקצועיים. על ידי שימוש בממשק פשוט, אפשרויות גמישות לניהול קבצים ואבטחה חזקה, המערכת שואפת לענות על הצרכים המגוונים של בסיס המשתמשים שלה ביעילות.

### **יעדים ומטרות:**

המטרה העיקרית של המערכת היא לספק פתרון אחסון ענן אמין, מאובטח וידידותי למשתמש לניהול קבצים דיגיטליים. כדי להשיג זאת, המערכת מתמקדת ביעדים הבאים:

אחסון קבצים מאובטח – המערכת מוודאת שמשתמשים יכולים לאחסן בבטחה את הקבצים והתיקיות שלהם מבלי לדאוג לגבי גישה לא מורשית או אובדן נתונים.

שיתוף ושיתוף קבצים – מאפשר למשתמשים לשתף קבצים בקלות עם אחרים, מה שמקל על שיתוף פעולה וחילופי מידע יעילים.

ארגון וניהול – מספק כלים למשתמשים לארגן את הקבצים שלהם בתיקיות, לשנות שמות של קבצים ותיקיות ולתחזק מערכת קבצים מובנית.

ניהול חשבון – מאפשר למשתמשים ליצור חשבונות, להתחבר בצורה מאובטחת, לשנות את שמות המשתמש שלהם ולמחוק חשבונות בעת הצורך.

פרטיות נתונים והצפנה – מגן על נתוני משתמש על ידי הצפנת כל התקשורת בין הלקוח לשרת, הבטחת פרטיות ואבטחה במהלך העברת הנתונים.

חוויה ידידותית למשתמש – ממשק פשוט ואינטואיטיבי כך שמשתמשים יוכלו לבצע משימות ביעילות, ללא קשר למומחיות הטכנית שלהם.

מדרגיות – המערכת מתוכננת להתמודדות עם צמיחה, הכלה של יותר משתמשים וכמויות גדולות יותר של נתונים לאורך זמן.

על ידי עמידה ביעדים אלו, המערכת שואפת לספק פלטפורמת אחסון ענן רב-תכליתית ומהימנה לשימוש אישי ומקצועי.

### **בעיות תועלת וחסכנות:**

ישנן כמה בעיות עיקריות שאותן אני שואף לפתור בפיתוח המערכת והמטרה הסופית היא ליצור מערכת שעובדת בצורה חלקה ללא חשש מאיבוד מידע או קריסה של התוכנה.

הבעיות העיקריות הינן:

1. ניהול המידע – המידע צריך להיות מנוהל בשרת באופן מאובטח שמאפשר רק לאנשים בעלי גישה לגשת לקבצים ולהבטיח לכל משתמש כי המידע שלו בטוח.
2. שיתוף מידע בצורה מאובטחת – כל משתמש צריך להיות בשליטה על הקבצים שלו ומי יכול לראות אותם ולערוך אותם. אין מצב בו קובץ של משתמש הגיע למשתמש אחר בצורה לא רצויה. הבטחון המיידעי הזה אינו יפגע ביכולות המשתמש.
3. שימוש פשוט – המערכת צריכה להיות פשוטה לשימוש ובעלת ממשק משתמש נוח ולא יותר מדי עמוס, וזה בזמן שהיא תהיה עשירה בפיצ'רים.

### **מערכות ותוכנות קיימות:**

1. **Google Drive:** אחסון קבצים בענן עם סנכרון למחשב ולנייד, שיתוף קבצים ותיקיות עם הרשאות (צפייה, עריכה), אינטגרציה עם Google Workspace (Docs, Sheets,

**יתרונות:** קלות שימוש, אחסון חינמי בנפח מוגבל (15GB)., אפשרויות מתקדמות לשיתוף ושיתוף פעולה בזמן אמת.

**חסרונות**: מגבלות נפח בחבילות החינמיות, תלות רבה באקו-סיסטם של Google.

1. **Dropbox:** אחסון, שיתוף וסנכרון קבצים. אפשרות לשיתוף קבצים עם קישורים מוגנים בסיסמה.

**יתרונות:** ממשק פשוט ואינטואיטיבי, שילוב טוב עם שירותי צד שלישי.

**חסרונות:** אחסון חינמי מוגבל (GB2), אין אפשרות לעריכה בזמן אמת כמו ב-Google Drive.

1. **OneDrive:** אינטגרציה עם ,Microsoft Officeאפשרויות שיתוף והרשאות מתקדמות.

**יתרונות:** חלק מחבילות Office 365 עם נפח אחסון גדול, אבטחת מידע ברמה גבוהה עם הצפנה.

**חסרונות:** תלות באקו-סיסטם של Microsoft -.

1. **:iCloud (Apple)** מיועד בעיקר למשתמשי Apple - , סנכרון נתונים בין מכשירי Apple – תמונות מסמכים וכו'...

**יתרונות:** פשטות למשתמשי Apple - , אינטגרציה מעולה עם macOS ו-iOS.

**חסרונות:** לא מתאים למשתמשי Windows או Android , נפח חינמי קטן (5GB) בלבד.

1. **Mega:** אחסון קבצים מוצפן מקצה לקצה, נפח אחסון חינמי גדול יחסית (20GB).

**יתרונות:** דגש חזק על פרטיות ואבטחה, אחסון חינמי בנפח גדול.

**חסרונות:** הממשק פחות ידידותי בהשוואה למתחרים, מהירות העלאה/הורדה מוגבלת בחבילות חינמיות.

**השוואה בין המערכת שלי למערכות הקיימות**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **תכונה** | **המערכת שלי** | **Google Drive** | **Dropbox** | **OneDrive** | **iCloud** | **Mega** |
| **אבטחה** | הצפנה מלאה בתעבורה | חלקי (TLS) | חלקי (TLS) | הצפנה חזקה | הצפנה בסיסית | הצפנה מקצה לקצה |
| **הרשאות מותאמות** | כן | כן | כן | כן | חלקית | כן |
| **אחסון חינמי** | 100GB | 15GB | 2GB | 5GB | 5GB | 20GB |
| **ממשק משתמש נוח וברור להבנה** | כן | חלקית | לא | לא | כן | פחות נוח |
| **שיתוף קבצים לא מוגבל** | כן | כן | כן | חלקי | חלקי | כן |
| **גישה ממכשירים שונים** | כן | כן | כן | כן | כן | כן |
| **מהירות חיבור מהירה ביותר** | כן | כן | כן | חלקי | כן | לא |

לסיכום, למערכת שלי ישנם יתרונות על פני מערכות קיימות אחרות והיא נועדה לייצר שימוש נוח ומועיל.

### **סקירת טכנולוגית:**

הפרויקט שלי אינו עושה בשימוש בטכנולוגיה חדשה ולא מוכרת, בעזרת שימוש בשפת תכנות פייתון (שהיא מהחדשניות בעולם ובעלת המון אופציות שימושיות) וניהול מסד נתונים בSQL, המערכת שלי תוכל לבצע את המטרה שלה, שכן קיימת כבר אך בעלת יתרונות על המתחרים, בצורה הכי יעילה מבחינת משאבים וכן נוחה למשתמש.

ישנם כמה סייגים שחשוב לציין במערכת שלי. ראשית המערכת נועדה לרוץ על שרת אחד, שכן כאשר המון משתמשים ירצו להתחבר שרת נוסף יצטרך להיפתח ולו לא יהיה את אותו מסד הנתונים. בנוסף לכך, נדרש שרת חזק מאוד ובעל אחסון רב מאוד בשביל שהמון משתמשים יוכלו לאחסן את המידע שלהם בו. עוד בעיה שנוצרת בשימוש במערכת שלי היא שהמערכת אינה כוללת Data Redundancy, כלומר אם קורה משהו לאחד מהכוננים – הקבצים שבו יאבדו ולא יהיה ניתן לשחזרם.

תיחום הפרויקט:הפרויקט שלי עוסק במגוון תחומים טכנולוגיים, תוך התמקדות בממשקים בין רשתות, מערכות הפעלה, וניהול נתונים. להלן תחומי העיסוק המרכזיים של המערכת, לצד התחומים שאינם מכוסים.

### **תחומים בהם המערכת עוסקת:**

1. **רשתות תקשורת -**

* **אבטחת תעבורה**: הצפנת נתונים בתעבורה באמצעות הצפנות (RSA, AES), מניעת גישה לא מורשית תוך שימוש במנגנוני אימות.
* **ניהול חיבורים בענן:** יצירת חיבור מאובטח בין לקוחות (Clients) לשרתים בענן, תמיכה בגישה ממכשירים שונים עם כתובת IP דינמית או סטטית.
* **ניהול שיתוף ברשת**: הפעלת מנגנונים לשיתוף קבצים מאובטח בין משתמשים ברשת, הגדרת הרשאות מבוססות תפקידים עבור משתמשים שונים.

1. **מערכות הפעלה -**

* **אינטגרציה בין מערכות הפעלה**: המערכת מאפשרת גישה מכל מערכות ההפעלה העיקריות: Windows, macOS, Linux, ואפילו iOS ו-Android באמצעות דפדפנים או אפליקציות.
* **ניהול קבצים ותיקיות**: מנגנון לארגון קבצים ותיקיות, דומה למערכת הקבצים במערכות הפעלה מקומיות, שימוש ב-Metadata לניהול מידע על הקבצים כמו גודל, שם, יוצר, ותאריך שינוי.
* **תמיכה בנפחים גדולים**: אפשרות לניהול אחסון בנפחים משמעותיים עם מיטוב ביצועי**.**

1. **ניהול נתונים במסד נתונים -**

* **אחסון מבוסס מסד נתונים**: שמירת מידע על קבצים, תיקיות, משתמשים, הרשאות, ושיתופים בבסיס נתונים רלציוני (SQL) או NoSQL.,ניהול גרסאות קבצים (File Versioning).
* **אופטימיזציה**: שמירת נתונים דחוסים להקטנת נפח האחסון הנדרש. מנגנוני גיבוי ושחזור נתונים במקרה של תקלה.

1. **אבטחת מידע -**

* **הצפנה מקצה לקצה**: שמירה על פרטיות המשתמשים על ידי הצפנה של מידע בזמן העלאה ואחסון.
* **אימות משתמשים**: מנגנונים כמו סיסמאות מוצפנות, אימות דו-שלבי (2FA), וניהול הרשאות מותאם אישית.
* **ניהול גישה מבוסס תפקידים (RBAC)**:ניהול הרשאות פרטני לפי קובץ, תיקיה, ומשתמש.

### **תחומים בהם המערכת לא מטפלת:**

1. **תחומי רשתות שאינם מכוסים -**

* **ניהול תשתית רשת**: המערכת אינה עוסקת בהגדרת שרתים, ניהול מתגים, או רכיבי חומרה אחרים של הרשת.
* **חיבוריות בין עננים (Multi-Cloud)**:המערכת אינה מציעה ניהול אחסון משולב בין ספקי ענן שונים (כגון Google Drive ו-AWS).
* **אופטימיזציה של מהירות תעבורה**: המערכת לא כוללת מנגנונים אוטומטיים להאצת העלאות או הורדות, כמו רשתות CDN.

1. **תחומי מערכות הפעלה שאינם מכוסים -**

* **שילוב ישיר עם מערכת הקבצים**: אין אפשרות להריץ את המערכת ישירות על מערכת קבצים מקומית (כמו FUSE ב-Linux).
* **גישה ללא אינטרנט**: המערכת מחייבת חיבור פעיל לאינטרנט ואינה מספקת פתרונות לגישה במצב לא מקוון (Offline).
* **שילוב עם אפליקציות צד שלישי**: המערכת אינה תומכת באופן מובנה בשילוב עם תוכנות כמו Office או Adobe Reader.

1. **תחומים עסקיים ותמיכת לקוח -**

* **שירותים מבוססי AI**: אין שימוש באלגוריתמים מבוססי בינה מלאכותית לניתוח נתונים (כגון ניתוח גודל שימוש, הצעות לניהול קבצים).
* **תמיכה טכנית אוטומטית**: המערכת אינה כוללת צ'אטבוט אוטומטי או פורטל תמיכה מתוחכם.

1. **תחומי אבטחה שאינם מכוסים -**

* **זיהוי איומים בזמן אמת**: המערכת אינה כוללת כלים לזיהוי מתקפות סייבר בזמן אמת.
* **מנגנוני אבטחה פיזיים**: לא מוצעים פתרונות לאבטחת שרתים פיזית או גיבויים חוץ-משרדיים.

המערכת מתמקדת בניהול קבצים ושיתוף מאובטח בענן, תוך שימוש בטכנולוגיות רשתות, אבטחת מידע, וניהול מערכות קבצים. עם זאת, היא אינה מתעסקת בתחומים טכניים עמוקים יותר כמו ניהול רשתות פיזיות, שילוב מערכות קבצים מקומיות, או אינטגרציות רחבות עם שירותי צד שלישי. תיחום זה מאפשר לי להתמקד ביעילות ובשיפור המערכת בנקודות החוזקה שלה.

## **פירוט תיאור המערכת (אפיון):**

### **תיאור מפורט של המערכת:**

המערכת שלי היא **פלטפורמת אחסון ושיתוף קבצים בענן** המיועדת למשתמשים פרטיים ועסקיים כאחד. היא נועדה להציע פתרון מודרני, מאובטח, וגמיש לניהול מידע דיגיטלי, תוך שילוב מנגנוני אבטחה מתקדמים וניהול משתמשים.

**מטרות המערכת**

1. **אחסון מאובטח של קבצים בענן**:
   * לאפשר למשתמשים לשמור את הקבצים שלהם בצורה בטוחה ונגישה מכל מכשיר המחובר לאינטרנט.
2. **שיתוף קבצים ותיקיות**:
   * מתן אפשרות לשיתוף קבצים ותיקיות עם משתמשים אחרים תוך שליטה על הרשאות השיתוף (קריאה, כתיבה, מחיקה וכו').
3. **ניהול יעיל של קבצים ותיקיות**:
   * לאפשר ארגון קבצים בצורה אינטואיטיבית עם אפשרויות לשינוי שמות, מחיקה, יצירת תיקיות חדשות, ושינויים נוספים.
4. **אבטחת מידע ושמירה על פרטיות**:
   * להבטיח שכל המידע המועבר והמאוחסן במערכת יישמר מוגן באמצעות מנגנוני הצפנה מתקדמים.
5. **ניהול משתמשים מותאם אישית**:
   * לספק ממשק לניהול חשבון משתמש, הכולל הרשמה, התחברות, שינוי פרטים אישיים, ומחיקת חשבון.

**תכונות עיקריות של המערכת**

1. **העלאה והורדה של קבצים**

* אפשרות להעלות קבצים למערכת ולשמור אותם בשרת מרוחק.
* הורדה קלה של קבצים למכשיר המקומי.
* הגבלת גודל העלאה לפי משתמש או תוכנית שימוש.

1. **יצירת תיקיות וניהולן**

* יצירת תיקיות לארגון מידע בצורה היררכית.
* שינוי שמות תיקיות ומחיקתן.
* ניווט קל בין תיקיות, כולל תצוגה מותאמת אישית (לדוגמה: תצוגת רשימה/תמונות).

1. **שיתוף קבצים ותיקיות**

* שיתוף קבצים ותיקיות עם משתמשים אחרים בתוך המערכת.
* הגדרת הרשאות:
  + קריאה בלבד.
  + קריאה וכתיבה.
  + מחיקה ושינוי שמות.
* יצירת קישורים זמניים לשיתוף חיצוני עם תוקף מוגבל.

1. **ניהול גרסאות קבצים**

* שמירת גרסאות קודמות של קבצים לשחזור שינויים שבוצעו.
* אפשרות להחזיר קובץ לגרסה ישנה במקרה של שגיאה.

1. **הרשאות מותאמות אישית**

* ניהול גישה לפי משתמשים או קבוצות משתמשים.
* הגדרת רמות גישה שונות עבור קבצים ותיקיות ספציפיים.

1. **אבטחת מידע**

* **הצפנה בתעבורה**:
  + כל המידע המועבר בין המשתמש לשרת מוצפן בפרוטוקולי TLS/SSL.
* **הצפנת מידע באחסון**:
  + שמירת קבצים בצורה מוצפנת בשרת, כך שאפילו למנהל המערכת אין גישה לתוכן.
* **מנגנון אימות דו-שלבי (2FA)**:
  + אפשרות להוסיף שכבת הגנה נוספת עם קוד אימות חד-פעמי.

1. **ניהול חשבון משתמש**

* הרשמה למערכת עם כתובת אימייל וסיסמה.
* שינוי שם משתמש וסיסמה לפי הצורך.
* מחיקת חשבון והסרת כל המידע השייך למשתמש.

1. **מעקב אחרי פעילות**

* יומן פעולות המאפשר למשתמש לראות אילו שינויים בוצעו (העלאות, מחיקות, שיתופים).
* אפשרות להפיק דוחות על שימוש בנפח אחסון.

1. **תמיכה בשפות**

* ממשק המשתמש תומך בשפות שונות (למשל עברית ואנגלית), כולל כיווניות נכונה לשפות ימין-שמאל (RTL).

**תהליכי עבודה מרכזיים**

**תהליך העלאת קובץ**

1. המשתמש בוחר קובץ מהמכשיר המקומי.
2. הקובץ עובר הצפנה בצד הלקוח (Client-Side Encryption).
3. הקובץ המוצפן נשלח לשרת ונשמר עם המידע הבא:
   * שם הקובץ.
   * גודל הקובץ.
   * מיקום בתיקיות.
   * הרשאות מוגדרות.

**תהליך שיתוף קובץ**

1. המשתמש בוחר קובץ או תיקיה לשיתוף.
2. המשתמש מגדיר את המוזמנים (כתובת אימייל או משתמש במערכת).
3. המשתמש קובע הרשאות גישה ומגבלות זמן לשיתוף.
4. נשלח קישור או הודעה על השיתוף למוזמנים.

**תהליך שחזור קובץ**

1. המשתמש נכנס לארכיון הגרסאות.
2. בוחר את הגרסה הרצויה לשחזור.
3. הקובץ מוחזר למיקום המקורי או נשמר כקובץ חדש.

**דגשים למערכת**

1. **התאמה אישית לצרכי משתמשים פרטיים ועסקיים**:  
   המערכת יכולה להתאים את עצמה לצרכים שונים על ידי קביעת גבולות אחסון, מנגנוני שיתוף, והרשאות מתקדמות.
2. **אבטחה ופרטיות בראש סדר העדיפויות**:  
   המערכת מוודאת שהמידע המאוחסן והמשותף יישאר חסוי, גם במקרה של פריצות או דליפות נתונים.
3. **תמיכה רב-פלטפורמית**:
   * גישה דרך דפדפן.
   * אפליקציות למכשירים ניידים.
   * אפשרות לסנכרון עם מחשבים שולחניים.

### **פירוט יכולות לכל משתמש:**

**משתמש רגיל (Standard User):**

* העלאה, הורדה, מחיקה, ועריכת קבצים ותיקיות.
* יצירת תיקיות וניהולן.
* שיתוף קבצים ותיקיות עם משתמשים אחרים.
* שינוי שם משתמש וסיסמה.
* צפייה במידע על חשבון, כמו נפח אחסון בשימוש.
* גישה להיסטוריית פעולות אישית.

**משתמש עם הרשאות שיתוף (Collaborator):**

* קבלת קבצים ותיקיות משותפים ממשתמשים אחרים.
* ניהול הרשאות על קבצים ותיקיות משותפים (בהתאם להרשאות שהוגדרו).
* עריכת קבצים ותיקיות משותפים.
* צפייה ומעקב אחר עדכונים בקבצים משותפים.

**מנהל צוות (Team Manager):**

* יצירה וניהול של קבוצות משתמשים.
* ניהול הרשאות גישה לקבצים ותיקיות של חברי הקבוצה.
* מעקב אחרי פעולות משתמשי הצוות (יומני פעילות).
* ניהול נפח אחסון לכל חבר צוות.
* הוספה או הסרה של משתמשים מהצוות.

**מנהל מערכת (Administrator):**

* ניהול משתמשים והרשאות ברמת המערכת.
* קביעת מגבלות אחסון עבור כל משתמש.
* ניטור פעילות כלל-מערכתית (יומני מערכת).
* טיפול בתקלות ושחזור נתונים.
* ניהול תשתית שרתים ותמיכה טכנית.

**משתמש זמני (Guest User):**

* גישה לקריאה בלבד לקבצים ותיקיות שהוגדרו כמשותפים.
* הורדת קבצים לפי הרשאה.
* אי-יכולת לשנות, למחוק, או להעלות מידע חדש.

**משתמש בתוכנית פרימיום (Premium User):**

* אחסון מורחב מעבר למגבלות המשתמש הרגיל.
* גישה לשירותי שחזור קבצים וגרסאות ישנות.
* שיתוף בלתי מוגבל של קבצים ותיקיות.
* הרשאות שיתוף חיצוניות ללא מגבלות.

### **פירוט הבדיקות המתוכננות:**

בדיקות הן חלק מהותי בתהליך הפיתוח על מנת להבטיח שהמערכת פועלת כהלכה ועונה על הדרישות שהוגדרו. להלן רשימת הבדיקות המתוכננות, כולל תיאור המטרה של כל בדיקה ואופן הביצוע שלה.

**1 - בדיקות פונקציונליות (Functional Testing)**

**העלאת קבצים**

* **מטרה**: לוודא שקבצים מועלים בהצלחה לשרת.
* **אופן ביצוע**:
  + העלאת קובץ מסוגים שונים (PDF, תמונה, וידאו).
  + בדיקה שהקובץ מופיע בממשק המשתמש.
  + בדיקת ההתנהגות כאשר מגבלת האחסון מתמלאת.

**הורדת קבצים**

* **מטרה**: לוודא שניתן להוריד קבצים מהשרת למחשב המקומי.
* **אופן ביצוע**:
  + הורדת קובץ שהועלה למערכת.
  + השוואת הקובץ שהורד עם המקורי (גודל וסוג קובץ).

**שיתוף קבצים ותיקיות**

* **מטרה**: לבדוק ששיתוף קובץ עובד בצורה תקינה.
* **אופן ביצוע**:
  + שיתוף קובץ עם משתמש אחר ובדיקת הגישה שלו.
  + שינוי הרשאות שיתוף (קריאה/כתיבה) ובדיקת השפעתן.

**ניהול תיקיות**

* **מטרה**: לוודא שניתן ליצור, למחוק, ולשנות שמות של תיקיות.
* **אופן ביצוע**:
  + יצירת תיקיות חדשות.
  + בדיקת הוספת קבצים לתיקיות והזזתם ביניהן.
  + בדיקת מחיקת תיקיות ריקות ומלאות.

**2 - בדיקות אבטחה (Security Testing)**

**אימות והתחברות (Login/Authentication)**

* **מטרה**: לוודא שמנגנון ההתחברות מאובטח.
* **אופן ביצוע**:
  + הזנת שם משתמש וסיסמה נכונים.
  + בדיקת התחברות עם סיסמה שגויה.
  + ניסיון התחברות מרובה ובדיקת נעילת החשבון.

**הצפנה בתעבורה**

* **מטרה**: לבדוק שהמידע המועבר בין המשתמש לשרת מוצפן.
* **אופן ביצוע**:
  + בדיקת שימוש בפרוטוקול TLS/SSL עם כל הבקשות.
  + בדיקת חשיפת מידע רגיש ברשת (בדיקת תעבורת HTTP באמצעות כלים כמו Wireshark).

**הרשאות גישה**

* **מטרה**: לוודא שמנגנון ההרשאות מונע גישה לא מורשית.
* **אופן ביצוע**:
  + משתמש רגיל מנסה לגשת לקבצים של משתמש אחר.
  + משתמש זמני מנסה לשנות או למחוק קובץ.

**3 - בדיקות ביצועים (Performance Testing)**

**בדיקות עומס (Load Testing)**

* **מטרה**: לבדוק את יכולת המערכת לטפל במספר רב של בקשות בו-זמנית.
* **אופן ביצוע**:
  + סימולציה של משתמשים רבים המעלים קבצים בו-זמנית.
  + מדידת זמן התגובה והיכולת של השרת להתמודד עם עומס.

**בדיקות מהירות העלאה והורדה**

* **מטרה**: לבדוק את זמני ההעלאה וההורדה תחת תנאי רשת שונים.
* **אופן ביצוע**:
  + העלאת והורדת קבצים בגודל שונה (לדוגמה, 1MB, 10MB, 1GB).
  + בדיקת השפעת מהירות חיבור נמוכה.

**4 - בדיקות שימושיות (Usability Testing)**

**ידידותיות ממשק משתמש (UI Testing)**

* **מטרה**: לוודא שהממשק נוח ואינטואיטיבי לשימוש.
* **אופן ביצוע**:
  + בדיקה שכל הפעולות נגישות ממשק המשתמש.
  + איתור אזורים מסורבלים או שאינם ברורים בממשק.

**תמיכה בשפות**

* **מטרה**: לוודא שהמערכת תומכת בשפות שונות ובכיווניות RTL/LTR.
* **אופן ביצוע**:
  + מעבר בין שפות שונות ובדיקת התצוגה שלהן.
  + בדיקת תמיכה בכיווניות ימין-שמאל בעברית או ערבית.

**5 - בדיקות התאוששות (Recovery Testing)**

**שחזור קבצים**

* **מטרה**: לוודא שניתן לשחזר קבצים שנמחקו או שונו.
* **אופן ביצוע**:
  + מחיקת קובץ ובדיקת השחזור מארכיון הגרסאות.
  + שיחזור גרסה קודמת של קובץ קיים.

**התמודדות עם כשלי מערכת**

* **מטרה**: לבדוק את התנהגות המערכת במקרים של כשל.
* **אופן ביצוע**:
  + כיבוי פתאומי של השרת במהלך העלאת קובץ.
  + בדיקת שלמות הקובץ לאחר הפעלה מחדש של השרת.

**סיכום**

תוכנית הבדיקות כוללת בדיקות מקיפות לכל היבט של המערכת: פונקציונליות, אבטחה, ביצועים, שימושיות, תאימות, התאוששות ותחזוקה. ביצוע שיטתי של כל הבדיקות יבטיח שמערכת האחסון והשיתוף בענן תעבוד באופן מיטבי ותעמוד בדרישות.

### **ניהול ותכנון לו"ז:**

**שלבי פיתוח מרכזיים ("אבנים גדולות") -**

1. **איסוף דרישות ותכנון ראשוני**
   * מטרה: להבין את הצרכים והדרישות מהמערכת, לתכנן ארכיטקטורה, ולבנות מפרט טכני.
   * זמן מתוכנן: 2 שבועות.
2. **תכנון ועיצוב ממשק משתמש (UI/UX)**
   * מטרה: יצירת אב-טיפוס לממשק, עיצוב חוויית משתמש ברורה ונוחה.
   * זמן מתוכנן: 2 שבועות.
3. **פיתוח צד שרת (Backend Development)**
   * מטרה: פיתוח ה-API, מסד הנתונים, והלוגיקה העסקית.
   * זמן מתוכנן: 4 שבועות.
4. **פיתוח צד לקוח (Frontend Development)**
   * מטרה: פיתוח ממשק המשתמש על פי העיצוב שהוגדר, כולל התממשקות עם ה-API.
   * זמן מתוכנן: 4 שבועות.
5. **אינטגרציה ובדיקות מערכת**
   * מטרה: חיבור כל רכיבי המערכת יחדיו וביצוע בדיקות פונקציונליות, ביצועים, אבטחה, ותאימות.
   * זמן מתוכנן: 3 שבועות.
6. **תיקונים ושיפורים**
   * מטרה: טיפול בבאגים ושיפור ביצועים בהתבסס על תוצאות הבדיקות.
   * זמן מתוכנן: 2 שבועות.
7. **פריסה וסיום פרויקט**
   * מטרה: פריסת המערכת בשרתים וכתיבת תיעוד מלא לשימוש ותחזוקה.
   * זמן מתוכנן: 2 שבועות.

**תכנון לוח זמנים ראשוני:**

|  |  |  |  |
| --- | --- | --- | --- |
| **שלב** | **תחילת עבודה** | **סיום מתוכנן** | **משך (שבועות)** |
| איסוף דרישות ותכנון ראשוני | שבוע 1 | שבוע 2 | 2 |
| תכנון ועיצוב ממשק משתמש | שבוע 3 | שבוע 4 | 2 |
| פיתוח צד שרת | שבוע 5 | שבוע 8 | 4 |
| פיתוח צד לקוח | שבוע 9 | שבוע 12 | 4 |
| אינטגרציה ובדיקות מערכת | שבוע 13 | שבוע 15 | 3 |
| תיקונים ושיפורים | שבוע 16 | שבוע 17 | 2 |
| פריסה וסיום פרויקט | שבוע 18 | שבוע 19 | 2 |

**לוח זמנים בפועל**

הלוח בפועל יתעדכן במהלך הפרויקט בהתאם להתקדמות, קשיים בלתי צפויים, ותעדוף משימות.

|  |  |  |  |
| --- | --- | --- | --- |
| **שלב** | **תאריך תחילה בפועל** | **תאריך סיום בפועל** | **משך (שבועות)** |
| איסוף דרישות ותכנון ראשוני | --- | --- | --- |
| תכנון ועיצוב ממשק משתמש | --- | --- | --- |
| פיתוח צד שרת | --- | --- | --- |
| פיתוח צד לקוח | --- | --- | --- |
| אינטגרציה ובדיקות מערכת | --- | --- | --- |
| תיקונים ושיפורים | --- | --- | --- |
| פריסה וסיום פרויקט | --- | --- | --- |

**מעקב והערכת לוח זמנים**

* **כלים למעקב:**
  + שימוש במערכת לניהול משימות ומעקב אחר התקדמות כל שלב.
  + עדכון שוטף של לוח הזמנים בפועל תוך כדי עבודה.
* **מדדים להצלחה:**
  + עמידה בזמנים או צמצום פערים במקרה של עיכובים.
  + השלמת כל שלב עם מסירה תקינה של תוצרים ועמידה בדרישות שהוגדרו.

**התמודדות עם עיכובים**

* חלוקה מחדש של משימות בין הצוותים.
* שימוש בפתרונות זמניים אם מתגלים חסמים בלתי צפויים.
* עדכון שוטף עם בעלי העניין בנוגע לשינויים בלוחות הזמנים.

### **ניהול סיכונים ודרכי התמודדות:**

**סיכון 1: עיכובים בפיתוח**

* **מהות הסיכון**: חלקים מהפיתוח עשויים להתעכב בשל בעיות טכניות, חוסר בתיאום בין צוותים, או מחסור במשאבים.
* **דרכים להתמודד**:
  + תכנון לוח זמנים ריאלי עם זמן "חיץ" למקרים לא צפויים.
  + חלוקה של משימות גדולות למשימות קטנות ומעקב אחר התקדמותן.
  + שימוש בכלים לניהול פרויקטים (Trello, Jira) כדי לשמור על סדר ומעקב.
  + אם מתגלה עיכוב, תגבור כוח אדם או העברת משימות לצוותים אחרים.
* **מה בוצע בפועל**:
  + עדכון לוח הזמנים בעקבות עיכובים טכניים בשלבי האינטגרציה.
  + הוספת מפתח נוסף לצוות הפיתוח במידת הצורך.

**סיכון 2: חוסר יציבות מערכת במהלך בדיקות**

* **מהות הסיכון**: המערכת עלולה לקרוס במהלך בדיקות עומס או פונקציונליות עקב באגים או תכנון שגוי.
* **דרכים להתמודד**:
  + ביצוע בדיקות יחידה (Unit Tests) מוקדם ככל האפשר על רכיבים מרכזיים.
  + ביצוע בדיקות אינטגרציה בשלבים מוקדמים כדי לאתר בעיות בחיבור בין חלקי המערכת.
  + שימוש בסביבות בדיקה מבודדות לפני העלאה לייצור.
* **מה בוצע בפועל**:
  + הטמעה מוקדמת של בדיקות אוטומטיות לזיהוי בעיות בשלב מוקדם.
  + תיקון באגים משמעותיים שהתגלו במהלך בדיקות עומס.

**סיכון 3: סיכוני אבטחה**

* **מהות הסיכון**: דליפת מידע אישי או פריצה לשרתים עלולות לפגוע באמון המשתמשים.
* **דרכים להתמודד**:
  + שימוש בפרוטוקולים מאובטחים (TLS/SSL) לכל התעבורה.
  + אחסון סיסמאות במאגר הנתונים בצורה מוצפנת (Hashing).
  + ביצוע סריקות אבטחה תקופתיות וזיהוי פרצות.
  + הגבלת הרשאות גישה לפי תפקידים.
* **מה בוצע בפועל**:
  + פריסת מנגנון הצפנת תעבורה מלא ובדיקת חדירות (Penetration Testing).
  + תיקון פרצות קטנות שהתגלו במהלך סריקות אבטחה ראשוניות.

**סיכום ניהול סיכונים**

הניהול כלל זיהוי מוקדם של סיכונים, הגדרת תהליכי התמודדות, וביצוע מעקב צמוד לאורך כל שלבי הפרויקט. תהליך זה עזר למזער את ההשפעה של סיכונים על איכות ולוחות הזמנים של המערכת.

# **תיאור תחום הידע**

## **פירוט מעמיק של יכולות המערכת:**

### **יכולות בצד שרת:**

**ניהול משתמשים**

**שם היכולת**: ניהול משתמשים  
**מהות**: ניהול מלא של משתמשים במערכת, כולל רישום, התחברות, שינוי פרטים, ושחזור סיסמה.  
**אוסף יכולות נדרשות**:

* יצירת חשבון חדש עם בדיקות לתקינות המידע (אימייל ייחודי, שם משתמש וסיסמה).
* אימות חשבון על ידי שליחת קוד אימות לאימייל.
* התחברות באמצעות אימות שם משתמש וסיסמה.
* שינוי פרטי משתמש, כולל שם משתמש וסיסמה.
* מנגנון לאיפוס סיסמה עם שליחת קוד זמני למייל.
* מחיקת חשבון תוך מחיקת כל המידע המשויך אליו מהמערכת.  
  **אובייקטים נחוצים**:
* ממשק משתמש ליצירת חשבונות ולניהול פרטי המשתמש.
* תקשורת בין הלקוח לשרת לפרוטוקולי התחברות ורישום.
* בסיס נתונים לאחסון פרטי המשתמשים.
* הצפנה לשמירה על פרטיות הסיסמאות והמידע האישי.

**ניהול קבצים ותיקיות**

**שם היכולת**: ניהול קבצים ותיקיות  
**מהות**: מתן אפשרות למשתמשים לבצע פעולות מגוונות על קבצים ותיקיות המאוחסנים במערכת.  
**אוסף יכולות נדרשות**:

* העלאת קבצים תוך שימוש במנגנון נעילה למניעת גישה כפולה.
* הורדת קבצים בשלבים (chunks) או כקובץ ZIP.
* מחיקת קבצים ותיקיות, תוך וידוא הרשאות מתאימות.
* שינוי שמות של קבצים ותיקיות.
* יצירת תיקיות חדשות.
* שיתוף קבצים ותיקיות עם משתמשים אחרים עם הרשאות מותאמות אישית.
* שחזור קבצים שנמחקו למיקומם המקורי.  
  **אובייקטים נחוצים**:
* ממשק משתמש לניהול קבצים ותיקיות.
* מנגנון נעילה למניעת התנגשות בגישה.
* בסיס נתונים לניהול מטה-נתונים של קבצים ותיקיות.
* מנגנון הרשאות מותאם אישית לכל משתמש.

**הרשאות וגישה**

**שם היכולת**: הרשאות וגישה  
**מהות**: שליטה מלאה על גישת משתמשים לקבצים ותיקיות במערכת.  
**אוסף יכולות נדרשות**:

* הגדרת הרשאות לכל קובץ ותיקייה (קריאה, כתיבה, מחיקה, שיתוף, הורדה).
* זיהוי משתמשים אורחים ומניעת פעולות לא מורשות.
* בקרת גישה מותאמת אישית לכל משתמש.  
  **אובייקטים נחוצים**:
* מנגנון הרשאות מבוסס נתונים.
* ממשק לניהול הרשאות עבור המשתמשים.

**ביצועים ואבטחה**

**שם היכולת**: ביצועים ואבטחה  
**מהות**: הבטחת אבטחת המידע וניהול משאבים יעיל במערכת.  
**אוסף יכולות נדרשות**:

* הצפנת התקשורת בין השרת ללקוח באמצעות RSA ומפתח משותף.
* מנגנון נעילה למניעת גישה כפולה לקבצים.
* הגבלת מהירויות העלאה והורדה לפי סוג המנוי.
* טיפול בניהול עומסים ואופטימיזציה של ביצועים.  
  **אובייקטים נחוצים**:
* מערכת הצפנה ופענוח.
* בסיס נתונים לניהול פרטי מנויים והגבלות.
* מנגנון לניהול משאבי מערכת.

**ממשק פרוטוקולים**

**שם היכולת**: ממשק פרוטוקולים  
**מהות**: ניהול פרוטוקולים מותאמים אישית לביצוע פעולות בין הלקוח לשרת.  
**אוסף יכולות נדרשות**:

* קבלת בקשות מהלקוח לפי קודים מותאמים (LOGIN, SIGNUP, UPLOAD וכו').
* ניתוח בקשות והעברתם למודול המתאים בשרת.
* החזרת תגובה מסודרת ללקוח.  
  **אובייקטים נחוצים**:
* מערכת תקשורת בין לקוח לשרת.
* מודול לעיבוד בקשות.

**ניהול מנויים**

**שם היכולת**: ניהול מנויים  
**מהות**: מתן שירותי אחסון מותאמים אישית לפי רמות מנוי שונות.  
**אוסף יכולות נדרשות**:

* הגדרת רמות מנוי (חינמי, בסיסי, פרימיום, מקצועי).
* ניהול מגבלות כמו אחסון ומהירות העלאה והורדה לפי רמת מנוי.
* שדרוג מנוי תוך שמירה על המידע הקיים.  
  **אובייקטים נחוצים**:
* בסיס נתונים לניהול פרטי מנויים.
* ממשק לניהול ושדרוג מנויים.

**תחזוקה וניהול מערכת**

**שם היכולת**: תחזוקה וניהול מערכת  
**מהות**: שמירה על פעילות תקינה של המערכת וניהול תקלות.  
**אוסף יכולות נדרשות**:

* רישום לוגים של כל הפעולות במערכת.
* ניטור תעבורה ושיפור ביצועים.
* מנגנון לשחזור אוטומטי של חיבורים שנקטעו.  
  **אובייקטים נחוצים**:
* מערכת ניטור ותיעוד.
* מודולים לשחזור ושיפור ביצועים.

**יכולות נוספות**

**שם היכולת**: יכולות נוספות  
**מהות**: תמיכה בפעולות מתקדמות להגדלת היציבות והאמינות של המערכת.  
**אוסף יכולות נדרשות**:

* שחזור נתונים שנמחקו בטעות.
* תיקון שגיאות ואופטימיזציה של תהליכים.  
  **אובייקטים נחוצים**:
* בסיס נתונים לגיבוי נתונים.
* מודול לניהול ותיקון שגיאות.

### **יכולות בצד לקוח:**

**ניהול משתמשים**

**שם היכולת**: ניהול משתמשים  
**מהות**: מתן אפשרות למשתמשים לנהל את חשבונותיהם ישירות מהאפליקציה.  
**אוסף יכולות נדרשות**:

* ממשק משתמש אינטואיטיבי לרישום, התחברות ושינוי פרטי חשבון.
* תהליך רישום עם קליטת פרטים ושליחתם לשרת.
* התחברות לחשבון קיים באמצעות אימייל וסיסמה.
* מנגנון לשחזור סיסמה הכולל שליחת קוד זמני לאימייל.
* הצגת הודעות למשתמש על הצלחה או כישלון של פעולות.  
  **אובייקטים נחוצים**:
* מסכי רישום והתחברות.
* חיבור מאובטח לשרת לשליחת פרטי המשתמש.
* מודול לניהול הודעות משתמש (success/error).

**ניהול קבצים ותיקיות**

**שם היכולת**: ניהול קבצים ותיקיות  
**מהות**: מתן אפשרות למשתמשים לנהל קבצים ותיקיות באחסון הענן דרך ממשק גרפי.  
**אוסף יכולות נדרשות**:

* הצגת רשימת הקבצים והתיקיות של המשתמש.
* העלאת קבצים ותיקיות לאחסון הענן.
* הורדת קבצים למחשב המקומי.
* יצירת תיקיות חדשות ישירות מתוך האפליקציה.
* שיתוף קבצים ותיקיות עם משתמשים אחרים על פי הרשאות.
* שינוי שמות של קבצים ותיקיות.
* מחיקת קבצים ותיקיות והעברתם לאזור "אשפה".  
  **אובייקטים נחוצים**:
* ממשק משתמש להצגת קבצים ותיקיות.
* חיבור לשרת לקבלת עדכונים בזמן אמת על תוכן המשתמש.
* מנגנון לניהול הרשאות שיתוף מתוך האפליקציה.

**הרשאות וגישה**

**שם היכולת**: הרשאות וגישה  
**מהות**: ניהול הרשאות שיתוף וגישה לקבצים ותיקיות.  
**אוסף יכולות נדרשות**:

* תפריט שיתוף קבצים עם הגדרת הרשאות (קריאה, כתיבה, מחיקה).
* הצגת מצב שיתוף לכל קובץ ותיקייה בממשק.
* שינוי הרשאות שיתוף בזמן אמת.
* מנגנון להסרת שיתופים קיימים.  
  **אובייקטים נחוצים**:
* מסך לניהול הרשאות.
* חיבור לשרת לעדכון הרשאות ותוכן.

**ביצועים ואבטחה**

**שם היכולת**: ביצועים ואבטחה  
**מהות**: שמירה על תקשורת מאובטחת עם השרת וניהול פעולות באופן יעיל.  
**אוסף יכולות נדרשות**:

* חיבור מוצפן לשרת לכל פעולה שמתבצעת דרך האפליקציה.
* ניהול זיכרון וטעינת תוכן בצורה אופטימלית.
* טיפול במקרים של חיבור איטי או ניתוק פתאומי.
* הצגת הודעות למשתמש על מצב התקשורת עם השרת.  
  **אובייקטים נחוצים**:
* מערכת הצפנה ופענוח נתונים.
* מודול לניהול חיבורים ותקלות.

**ממשק פרוטוקולים**

**שם היכולת**: ממשק פרוטוקולים  
**מהות**: ניהול הבקשות והתשובות בין הלקוח לשרת בצורה מסודרת.  
**אוסף יכולות נדרשות**:

* שליחת בקשות לשרת בפורמט אחיד ומאובטח.
* קבלת תשובות מהשרת וניתוחן.
* טיפול במקרי שגיאה והצגתן למשתמש בצורה ברורה.  
  **אובייקטים נחוצים**:
* מודול לשליחת בקשות וקבלת תשובות.
* ממשק לטיפול בשגיאות ותקלות.

**ניהול מנויים**

**שם היכולת**: ניהול מנויים  
**מהות**: הצגת פרטי המנוי של המשתמש ואפשרות לשדרוג.  
**אוסף יכולות נדרשות**:

* הצגת פרטי המנוי הנוכחי (אחסון זמין, מהירות העלאה והורדה).
* מתן אפשרות לשדרוג מנוי מתוך האפליקציה.
* ניהול תשלומים ומעקב אחר תקופת המנוי.  
  **אובייקטים נחוצים**:
* מסך פרטי מנוי ושדרוג.
* חיבור לשרת לעדכון סטטוס המנוי.
* מודול לניהול תשלומים.

**תחזוקה וניהול מערכת**

**שם היכולת**: תחזוקה וניהול מערכת  
**מהות**: שמירה על תקינות האפליקציה בצד הלקוח וניהול תקלות.  
**אוסף יכולות נדרשות**:

* טיפול במקרי קריסה ושמירה על יציבות האפליקציה.
* עדכון גרסאות האפליקציה באופן אוטומטי.
* מנגנון לדיווח על תקלות והצעת פתרונות למשתמש.  
  **אובייקטים נחוצים**:
* מודול לניהול עדכונים.
* ממשק לדיווח תקלות.
* חיבור לשרת לקבלת מידע על תקלות ועדכונים.

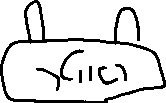
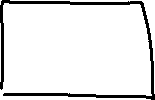
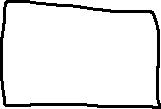
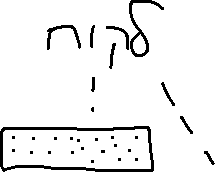
# **מבנה/ארכיטקטורה של המערכת**

## **תיאור הארכיטקטורה של המערכת:**

### **תיאור החומרה:**

החומרה במערכת שלי מורכבת בבסיס משרת שרץ על מחשב ווינדוס כלשהו ומאחסן את הקבצים שמעלים אליו המשתמשים, כלומר השרת דורש מחשב מתפקד עם זיכרון, בנוסף אם נדרשת התחברות של לקוחות מרשת אחרת, יש לאפשר Port Forwarding במחשב השרת.

הלקוח גם הוא ירוץ על מחשב ווינדוס, אך בניגוד לשרת הלקוח הוא בעל ממשק משתמש גרפי, כלומר בכדי להשתמש במערכת, הלקוח יצטרך גם מקלדת ועכבר בשביל לנוע ולהשתמש בממשק. הלקוח לא צריך הרשאות מיוחדות בכדי להתחבר לשרת, גם כזה שלא נמצא באותה רשת פנימית.



### **שרטוט:**

## **טכנולוגיה רלוונטית:**

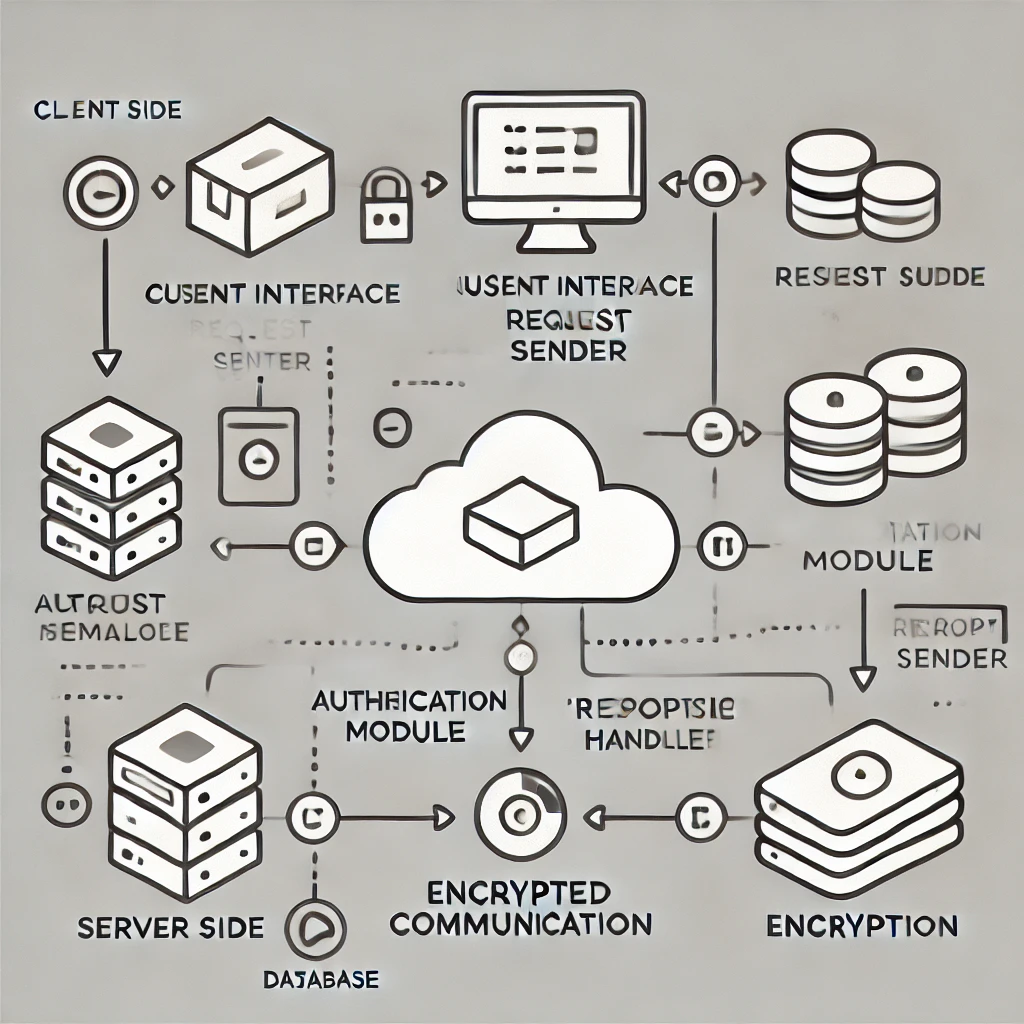
המערכת שלי בנויה בשפת התכנות פיתון, בחרתי לתכנת בפיתון מכמה סיבות. סיבה ראשונה היא שאני שולט בפיתון ומכיר את השפה טוב כך שאני יכול לבנות פרויקט ברמה גבוהה ולהצליח לפתור בעיות בקלות. בנוסף בפיתון יש המון ספריות שימושיות שעוזרות עם בניית הפרויקט. השתמשתי בספרייה PYQT6 לממשק המשתמש שלי מכיוון שהיא נוחה לשימוש, יש לה תוכנת עיצוב דפים (Designer) ואפילו ניתן להוסיף עיצוב עם CSS.

המערכת שלי רצה על מערכת ההפעלה ווינדוס. בחרתי לכתוב את המערכת כך שתתאים ווינדוס מכיוון שזוהי מערכת ההפעלה הנפוצה ביותר בעולם וכך תהיה גישה לקהל רחב יותר של משתמשים. בנוסף לכך, ישנם אמולטורים במערכות הפעלה אחרות כגון לינוקס ומאק אואס, שבעזרתם ניתן להריץ תוכנות ווינדוס כך שגם משתמש שאינו ווינדוס יוכל להשתמש במערכת.

בשביל התקשורת השתמשתי בתקשורת סוקטים אסינכרונית. כלומר השרת מפעיל סוקט שמאזין לבקשות, בכל פעם שמגיעה בקשה הוא מעבד אותה ומחזיר תשובה למקור. הלקוח עובד בצורה דומה, הוא מאזין כל הזמן לתשובות, וכאשר המשתמש מבקש, הוא שולח בקשות לשרת. התקשורת האסינכרונית תורמת לכך שבזמן שהלקוח מעלה קובץ גדול, הממשק לא יתקע והוא יוכל להמשיך לבקש דברים מהשרת, הוא לא יתקע עד לתשובה מהשרת.

אני בחרתי בפרויקט הזה גם מתוך תחומי העניין האישיים שלי. אני חושב שמערכת ענן היא דבר שימושי מאוד ומגניב, וביצירת מערכת כזו בעצמי אני יכול להוסיף פיצ'רים ואופציות מיוחדות שנראות לי מתאימות שלא בהכרח יש במערכות אחרות. בנוסף אני חושב שזה גם מאוד שימושי לשימוש האישי שלי, מערכת אחסון טובה ומהירה שלי אין בה הגבלה.

## **זרימת המידע במערכת:**



## **תיאור האלגוריתמיקה בפרויקט:**

### **ניסוח וניתוח של הבעיה האלגוריתמית:**

**תיאור אלגוריתמיקה של הפרויקט**

**ניסוח הבעיה האלגוריתמית**

מערכת האחסון בענן עוסקת בניהול נתונים בצורה מאובטחת, יעילה ומבוזרת. הבעיה המרכזית שיש לפתור היא כיצד לספק למשתמשים גישה נוחה ושקופה לקבצים ותיקיות תוך הבטחת אבטחת מידע, ביצועים גבוהים ותמיכה בריבוי משתמשים בו זמנית. הבעיה מתפרקת למספר תתי-בעיות אלגוריתמיות עיקריות:

1. **ניהול משתמשים והרשאות**: כיצד לאמת משתמשים ולנהל את הרשאותיהם באופן דינמי.
2. **אחסון נתונים מבוזר ומאובטח**: כיצד לאחסן נתונים בצורה שמבטיחה גם גישה מהירה וגם הצפנה.
3. **תמיכה בעומסים גבוהים**: כיצד לנהל העלאות והורדות בו-זמנית עבור מספר רב של משתמשים.
4. **שיתוף קבצים**: כיצד ליישם מנגנון גמיש לשיתוף קבצים בין משתמשים עם בקרת הרשאות ברזולוציה גבוהה.

**ניתוח הבעיה האלגוריתמית**

* 1. **ניהול משתמשים והרשאות**
* **בעיה**: אימות זהות המשתמשים ושמירת נתונים רגישים כגון סיסמאות באופן מאובטח.
* **פתרון אלגוריתמי**: שימוש בהצפנה עם bcrypt לשמירת סיסמאות והצפנת תקשורת באמצעות RSA להחלפת מפתחות ו-AES לתעבורת נתונים.
  1. **אחסון נתונים מבוזר ומאובטח**
* **בעיה**: שמירה על זמינות גבוהה של נתונים תוך שמירה על פרטיות.
* **פתרון אלגוריתמי**: שימוש בבסיס נתונים מבוסס sqlite3 לניהול מטה-נתונים, ואחסון פיזי של קבצים במערכת קבצים עם מנגנוני גיבוי והצפנה.
  1. **תמיכה בעומסים גבוהים**
* **בעיה**: טיפול בו זמנית בבקשות העלאה והורדה של משתמשים רבים.
* **פתרון אלגוריתמי**: יישום תורים אסינכרוניים (Asynchronous Queues) לניהול משימות ברקע, חלוקת עומסים דינמית (Load Balancing) וחיתוך קבצים לחתיכות קטנות (Chunks) כדי לשפר את ביצועי ההעברה.
  1. **שיתוף קבצים**
* **בעיה**: ניהול שיתופים דינמיים עם הרשאות שונות (קריאה, כתיבה, מחיקה וכו').
* **פתרון אלגוריתמי**: ניהול טבלת הרשאות בבסיס הנתונים, ושימוש במנגנון היררכי לבדיקה האם למשתמש יש גישה ישירה או עקיפה (לדוגמה, דרך שיתוף).

**דרישות ביצועים**

* **אבטחת מידע**: זמן חישוב הצפנה/פענוח קצר, תוך עמידה בתקני אבטחה מודרניים.
* **סקלאביליות**: המערכת צריכה לתמוך באלפי משתמשים פעילים בו זמנית עם גישה מהירה לנתונים.
* **זמינות גבוהה**: המערכת צריכה להבטיח מינימום השבתות, עם מנגנוני התאוששות מהירה מתקלות.

### **אלגוריתמים קיימים:**

**1. הצפנה ואימות משתמשים**

* **bcrypt**: אלגוריתם להצפנת סיסמאות על ידי שימוש במנגנון Hashing משופר המוסיף Salt לכל סיסמה כדי למנוע התקפות מילון.
* **RSA + AES**: RSA משמש להחלפת מפתחות בצורה מאובטחת, ולאחר מכן AES משמש להצפנה סימטרית של התקשורת בין הלקוח לשרת.

**2. אחסון נתונים**

* **SQLite**: מנוע בסיס נתונים קל משקל המאפשר ניהול מטה-נתונים של קבצים ותיקיות בצורה מאובטחת ויעילה.
* **מערכת קבצים מבוזרת**: שימוש במנגנונים לניהול הקבצים הפיזיים כולל גיבוי, אחסון מוצפן וניהול גרסאות.

**3. טיפול בעומסים גבוהים**

* **Asynchronous Task Queues**: ניהול משימות ברקע בצורה אסינכרונית עם תורים כדי למנוע עומסים על השרת.
* **Load Balancing**: חלוקת עומסים בין משאבים שונים במערכת כדי לשפר את הביצועים.

**4. מנגנוני הרשאות ושיתוף**

* **RBAC (Role-Based Access Control)**: ניהול הרשאות על בסיס תפקידים המאפשר גמישות בניהול גישה לקבצים ותיקיות.
* **טבלאות הרשאות בבסיס הנתונים**: כל קובץ או תיקייה מקושרים לרשימת הרשאות מותאמות אישית לכל משתמש.

### **סקירת הפתרון הנבחר:**

הפתרון הנבחר משלב מספר אלגוריתמים קיימים לצד התאמות ייחודיות שנעשו לצורך מימוש המערכת.

**נימוק הבחירה בפתרון**

1. **שימוש באלגוריתמים מוכרים ומאובטחים:**
   * bcrypt להצפנת סיסמאות מספק שכבת אבטחה חזקה נגד התקפות brute-force.
   * שילוב של RSA ו-AES מאפשר תקשורת מוצפנת מאובטחת מבלי לפגוע בביצועים.
2. **שימוש במנגנון מבוזר לניהול עומסים:**
   * חלוקת הנתונים למקטעים קטנים (Chunks) מאפשרת עבודה במקביל ומשפרת ביצועים בעת העלאה והורדה.
   * Load Balancing מונע עומסים על השרת ומבטיח חוויית משתמש חלקה.
3. **ניהול הרשאות גמיש ויעיל:**
   * RBAC (Role-Based Access Control) מאפשר קביעת הרשאות מותאמות אישית לכל משתמש.
   * ניהול טבלאות הרשאות בבסיס הנתונים מספק פתרון גמיש לניהול שיתופי קבצים.

**שלילת פתרונות אלטרנטיביים**

1. שימוש בגיבוי קבצים בלבד ללא בסיס נתונים: פתרון זה היה מגביל את יכולת ניהול המטה-נתונים של הקבצים ולא מאפשר ניהול הרשאות מתקדם.
2. שימוש בהצפנה סימטרית בלבד: פתרון זה היה דורש העברת מפתחות פרטיים בין הלקוח לשרת, דבר שהיה מסכן את אבטחת המידע.
3. ניהול תהליכים סינכרוניים בלבד: היה גורם לעיכובים משמעותיים בעת העלאה והורדה של קבצים ולא מאפשר ביצועים גבוהים.

**דרישות ביצועים**

* אבטחת מידע: זמן חישוב הצפנה/פענוח קצר, תוך עמידה בתקני אבטחה מודרניים.
* סקלאביליות: המערכת צריכה לתמוך באלפי משתמשים פעילים בו זמנית עם גישה מהירה לנתונים.
* זמינות גבוהה: המערכת צריכה להבטיח מינימום השבתות, עם מנגנוני התאוששות מהירה מתקלות.

## **תיאור סביבת הפיתוח:**

### **כלי הפיתוח הנדרשים:**

הפרויקט נבנה בשפת Python, המספקת גמישות רבה ונוחות פיתוח, יחד עם ספריות מתקדמות לאבטחת מידע, ניהול תקשורת ועבודה עם ממשקי משתמש. לצורך פיתוח וניהול הקוד, נעשה שימוש בסביבת העבודה Visual Studio Code, המאפשרת עריכה נוחה, אינטגרציה עם בקרת גרסאות, ותמיכה בתוספים שימושיים לשיפור חוויית הפיתוח. Visual Studio Code מציע תמיכה רחבה בכתיבת קוד Python עם כלי בדיקה, ניפוי שגיאות, ותוספים לניתוח ביצועים. בנוסף, פלטפורמת הפיתוח כוללת את ספריות PyQt6 ליצירת ממשקי משתמש, SQLite לניהול בסיס הנתונים, ו-bcrypt ו-RSA לאבטחת מידע. בחירת הכלים הללו נבעה מהצורך במערכת אמינה, מאובטחת וגמישה המסוגלת להתמודד עם דרישות מורכבות כמו הרשאות, שיתוף קבצים ותעבורה מוצפנת.

### **הסביבה והכלים הנדרשים לבדיקות:**

לצורך בדיקות המערכת, נעשה שימוש בקובץ **log**  לתיעוד פעולות קריטיות, איתור בעיות וניטור תקלות במהלך הריצה. כמו כן, שולבו **נקודות עצירה (breakpoints)** במהלך הפיתוח כדי לאפשר ניפוי שגיאות ידני ובדיקת זרימת הקוד בזמן אמת. הבדיקות בוצעו באמצעות בדיקות ידניות לכל אחת מהפונקציות המרכזיות, כולל אימות משתמשים, העלאת והורדת קבצים, ניהול הרשאות ושיתוף קבצים, כדי לוודא שכל רכיב פועל כמצופה.

בנוסף, נעשה שימוש ב-**GitHub** לניהול גרסאות, שמירת שינויים ובידוד משתנים חשובים, מה שמאפשר חזרה לגרסאות קודמות במקרה של תקלה. כך ניתן היה לבדוק כל שינוי באופן מבוקר לפני שילובו בקוד המרכזי.

## **פרוטוקול התקשורת:**

### **תיאור מילולי של הפרוטוקול:**

מבנה הפרוטוקול במערכת מבוסס על מבנה נתונים אחיד וקבוע, המאפשר תקשורת אמינה בין הלקוח לשרת. כל הודעה מורכבת ממספר חלקים עיקריים, המסודרים בפורמט קבוע, על מנת להבטיח עיבוד יעיל וקריא של הבקשות והתגובות.

הפרוטוקול בנוי כך שה**4 הבתים הראשונים** מוקדשים לאורך ההודעה, ומיוצגים כנתונים בינאריים. הדבר מאפשר לשרת וללקוח לדעת מראש כמה נתונים עליהם לקרוא מהזרם, ובכך למנוע קריאות חסרות או קריאות עודפות של מידע.

לאחר מכן, מופיע **קוד ההודעה**, אשר מורכב מ-**4 תווים גדולים באנגלית** (לדוגמה: "LOGN", "UPLD" , "DELT"). קוד זה מאפשר לשרת לזהות במהירות את סוג הבקשה, ומקל על ניתוח ההודעה.

לאחר קוד ההודעה, מגיעים **השדות השונים של ההודעה**, כאשר לכל סוג הודעה יש כמות שונה של שדות בהתאם לצרכיה. השדות מופרדים זה מזה באמצעות התו | , המאפשר פענוח פשוט ומהיר. לדוגמה, הודעת התחברות ("LOGN") עשויה לכלול את שם המשתמש והסיסמה כשדות, ואילו הודעת העלאת קובץ ("UPLD") תכלול שם קובץ, גודל, ונתוני ההצפנה. בכל הודעה קודם כל מופיעים שדות החובה ולאחר מכן שדות הרשות.

מבנה זה מבטיח שהשרת והלקוח מסוגלים לתקשר בצורה עקבית ויעילה, תוך שמירה על גמישות בהוספת פקודות חדשות בעתיד.

### **פירוט כלל ההודעות הזורמות במערכת:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **שם ההודעה** | **קוד ההודעה** | **כיוון** | **שדות חובה** | **שדות רשות** |
| התחברות | LOGN | לקוח - שרת | שם משתמש/אימייל, סיסמה | שמירת התחברות זמנית |
|  |  |  |  |  |

## **מסכי המערכת:**

### **תיאור כל מסך:**

### **תרשים מסכים:**

## **מבני הנתונים:**

### **פירוט מבני הנתונים:**

### **פירוט מאגרי המידע של המערכת:**

### **מסד נתונים:**

## **סקירת חולשות ואיומים:**

# **מימוש הפרויקט**

## **סקירת כל המודולים והמחלקות המרכיבים את המערכת והקשרים ביניהם:**

### **מודולים מיובאים:**

### **מודולים שפתחתי:**

## **קטעי קוד ופיתוחים מיוחדים:**

## **מסמך בדיקות מלא:**

### **בדיקות שתוכננו בשלב האפיון:**

### **בדיקות נוספות:**

# **מדריך למשתמש**

## **קבצי המערכת:**

## **התקנת המערכת:**

### **הסביבה הנדרשת:**

### **הכלים הנדרשים:**

### **מיקומי קבצים:**

### **נתונים התחלתיים:**

### **רשת:**

### **ארכיטקטורה נדרשת:**

## **משתמשי המערכת:**

### **אופן הפעלת המשתמש:**

### **צילומי מסכי הפרויקט:**

# **רפלקציה**

## **תהליך העבודה:**

## **תהליך הלמידה:**

## **כלים נלקחים להמשך:**

## **תובנות מהתהליך:**

## **ראייה לאחור:**

## **איך הייתי משפר:**

## **שאלות חקר עצמי:**

## **תודות:**

# **ביבליוגרפיה**

# **נספחים**