

## SyncSphere



**פרויקט גמר בהגנת סייבר ומערכות הפעלה**

**איל פיקז**

**שם המורה - צבי שירן**

**תאריך הגשה - 29.04.2025**



<b>מבוא:</b>	<b>3</b>
ייזום:	3
הגדרת הלוקוט:	3
הגדרת יעדים:	4
סקירה פתרונות קיימים:	4
סקירה טכנולוגיות הפרויקט:	5
תיכון הפרויקט:	5
פירוט תיאור המערכת:	5
תיאור המערכת:	5
פירוט יכולות של המערכת:	6
פירוט הבדיקות שמתוכננות לפרויקט:	6
תכנון וניהול לוח זמנים:	9
ניהול הסיכונים בפרויקט:	10
<b>תיאור תחום הידע:</b>	<b>11</b>
צד לקוחות:	11
<b>מבנה/ארქיטקטורה של הפרויקט:</b>	<b>17</b>
תיאור הארכיטקטורה של המערכת המוצעת:	17
תיאור החומרה:	17
תיאור הטכנולוגיה הרלוונטית:	18
תיאור זרימת המידע במערכת:	19
תיאור האלגוריתמים במערכת:	29
תיאור סביבת הפיתוח:	30
תיאור פרוטוקול התקשרות:	30
תיאור מסכי המערכת:	31
תיאור מבני הנתונים:	35
סקירה חולשות ואיומים:	37
<b>ימוש הפרויקט:</b>	<b>38</b>
סקירה המודולים, המחלקות וकשרי הגומלין ביניהם:	38
סקירה המודולים המיבאים:	38
מודולים שפיתחתי:	38
בעיות אלגוריתמיות - פירוט:	43
תיאור הבדיקות במערכת:	50
<b>מדריך למשתמש:</b>	<b>55</b>
פירוט כל קבצי המערכת:	55
התקנת המערכת:	57
הפעלת המערכת:	57
<b>סיכום אישי - רפלקציה:</b>	<b>1</b>
<b>ביבליוגרפיה:</b>	<b>1</b>
<b>נספחים:</b>	<b>1</b>
קוד הפרויקט:	1
app.py:	1
auth.py:	1
config.py:	1
file_management_routes.py:	1

file_management.py:	1
friend_managment_routes.py:	1
friend_management.py:	1
models.py:	1
sync.py:	1
grand_server.py:	1
templates/dashboard.html:	1
templates/friends_files.html:	1
templates/friends_list.html:	1
templates/incoming_requests.html:	1
templates/login.html:	1
templates/register.html:	1
static/css/auth.css:	1
static/css/dashboard.css:	1
של הפרויקט: <a href="#">github</a> -קישור לעמודה	1

## מבוא:

## יעזום:

הפרויקט שבחרתי לבצע הוא מערכת לגיבוי קבצים, שמירתם ושיתופם עם משתמשים אחרים במערכת. המוצר מאפשר למשתמש לאחסן קבצים על גבי שרת מרוחק ולהורידם במידת הצורך. המערכת מורכבת ממספר שירותים אזוריים (שאיתם המשתמש יוצר אוינטראקטיבי) ושרת אחד מרכזי האחראי על סנכרון הגיבויים בין השירותים האזוריים. בחרתי בפרויקט זה כיון שפרויקט זה שם דגש על עבודה עם מערכת הקבצים של המחשב, תחום שאוטי אישית מעוניין ושהאני רוצה לעסוק בו. האתגרים שאני צופה בפרויקט הם:

1. מימוש משקל המשתמש(עליו ארכיטיב בהמשך).
2. ניהול עבודה עם מסד הנתונים בצורה חכמה ויעילה.
3. ניצול מערכת הקבצים בצורה מקסימלית ועובדת כמה שיותר נוחה.
4. ניהול שיתוף מידע בין שירותים אזוריים
5. מחשבה על בעיות - כמו, מה קורה במידיה והשרות קורוט.
6. ניהול הסנכרון בין השירותים האזוריים כך שפעולה המתבצעת בשרת אזורי תתבצע גם בשאר השירותים.
7. למידה עצמית של הנושאים הרלוונטיים.

## **הגדרת הלקוח:**

המערכת מיועדת לאנשים שמחפשים פתרון מאובטח וידידותי למשתמש לניהול גיבוי קבצים ושיתופם עם משתמשים ספציפיים. המערכת מותאמת במיוחד לשיתוף קבצים בין שני אנשים שונים. אפשר לפעול עם המערכת ביותר מכך, אך פחות מומלץ לעשות זאת והמערכת אינה פועלת בצורה

אידיאלית. בנוסף לכך, המערכת יכולה להיות פתרון סבירת עבודה לחברה/צוות שרצה לחלוק בין חברי הצוות השונים.

### הגדרת יעדים:

היעדים המרכזיים שהצבתי לעצמי בתחום הפרויקט הם:

1. על המערכת לאפשר גיבוי של קבצים על שרת מרוחק, בנוסף למספר פעולות בסיסיות אחרות.
2. על המערכת לספק אפשרות להוספת חברים וצפיה בקבצייהם.
3. על המערכת לספק פתרון למקרים שבו שרת של המערכת קורס/לא מתפרק.
4. פשוטות - על המערכת להיות ידידותית למשתמש ופושטה להבנה כך שגם אנשים שאינם משתמשים בטכנולוגיה על בסיס קבוע יכולו להשתמש בצורה קלה יחסית.
5. אבטחה - על המערכת לאחסן את הקבצים של המשתמש בצורה בטוחה ומוצפנת כך שלא תתאפשר גישה לקבצים אלו. בנוסף לכך, על המערכת לשמר נתונים רגילים כגון סיסמאות בצורה מאובטחת שבמידה ונתונים אלו יודלפו, נתונים אלו לא יפגעו מהדיפה כזו.
6. זמינות - על המערכת להיות נגישה בכל זמן שהוא ובכל מקום שהוא, אף במקרה של קריוסות של השירותים/בעיות אחרות לאפשר גישה למערכת.

היעדים האופציונליים שהצבתי לעצמי בתחום הפרויקט הם:

1. הוספת אפשרות לניהל כמה גרסאות של קובץ אחד.
2. הוספת אפשרות לשילוח הودעות בין חברים.
3. אסתטיות - על המערכת להיות ידידותית לעין.

### בעיות תועלות וחסרונות:

הבעיה שהמערכת רצה לספק פתרון היא אפשרות לגיבוי קבצים והעברה שלהם ממחשב למחשב במידה הצורך. בנגד לשירותי גיבוי קבצים אחרים, על המערכת לספק אפשרות גם לשף קבצים אלו עם משתמשים אחרים בצורה פשוטה וידידותית למשתמש.

1. על המערכת לספק פתרון פשוט לגיבוי קבצים.
2. על המערכת לספק אפשרות לשף גישה לקבצים הנשמרים למשתמשים אחרים, ובמידת הצורך לשף גישה רק לךם.
3. על המערכת לסייע עם עצמה את השינויים המתבצעים בכל שרת ועדכן מסד הנתונים במקרה של כל פעולה מרכזית במערכת(עליהן ארchip בmarsh).
4. על המערכת לשמור על קבצי ונתוני המשתמש בצורה מאובטחת.

### פתרונות קיימים:

פתרונות קיימים הדומים לפרויקט שלי הם:

1. Google drive - כלי של גугл לשימור קבצים ושיתופם עם אנשים אחרים במידה הצורך. השימוש המركזי בין הפרויקט שלי לכלי זה הוא שכלי זה מבוסס יותר לעבודה עם מוצריהם של

גוגל ומותאם יותר לשירותי גוגל(כגון מסמכים של גוגל, ג'מייל ועוד) בעוד שהפרויקט שלו עומד בפני עצמו ומספק פתרונות כללים גם לקבצי מערכת שאינם קשורים לוגול.

2. One drive - כלי המאפשר העלאת, הורדת קבצים ושיתופם עם משתמשים אחרים. השוני - המרוצי בין הפרויקט שלו לכלי זה הוא שכלி זה שמו דגש על עבודה עם המערכת ויינידוז עם הכלים של מיקרוסופט אופיס. בנגדו לכך, הפרויקט שלו אינו ממוקד על מערכת הפעלה ספציפית והמשתמש לא באמת יחווה הבדל מרכזיא אם הוא יגיש לפרויקט שלו מערכת macos או kali המבוסס על גיט לשמרות קבצים וניהול גרסאות באמצעות גיט. השוני העיקרי - github בין כל זיה לבין הפרויקט שלו הוא שכלி זה מתמקד בעיקר בניהול גרסאות ובקבצי קוד במירוץ. בנגדו לכך, הפרויקט שלו מותאם לקבצים מסוימים שונים ולא מתעדף/מתמקד באחד.

ניתן לראות שרוב התחלופות הפופולריות מתמקדות בנישה מסוימת. בנגדו לכך, הפרויקט שלו מותאם לכל הנישות השונות במידה שווה. אמנם הפרויקט שלו אינו ברמה גבוהה בכל נישה כגון התחלופות שהצעתי פה, אך הפרויקט שלו עשוי להיות אלטרנטיבה לפתרונות אלו בכל אחד.

## **סקירת טכנולוגיות הפרויקט:**

הפרויקט שלו עוסק בטכנולוגיה קיימת. הטכנולוגיות העיקריות שהפרויקט שלו עוסק בהן הן יರחבי בהמשך.

לפרויקט ישן מספר הגבלות שצריך לזכור בחשבון על מנת להשתמש בפרויקט:  
1. על המחשב להיות בעל חיבור לאינטרנט.  
2. למראות הפשטות היחסית של הפרויקט, מומלץ לעיין במדריך למשתמש.

### **תיכון הפרויקט:**

#### תוכנה:

1. כתיבה בשפה פיתון
2. כתיבה בשפות css, html על מנת לעבוד עם ממשק משתמש.
3. הצפנות.

4. תקשורת באמצעות SoketIP(שרת מרובה לקוחות).

#### אחר:

5. עבודה עם מסד נתונים בשימוש SQLite3
6. עבודה עם מערכת הקבצים של המחשב.

## **פירוט תיאור המערכת:**

### **תיאור המערכת:**

המערכת מורכבת מספר חלקים עליהם אפרט מיד:

1. שרת אזרוי - שרת Aynıו המשמש מבצע אינטראקטיבי. השרת האזרוי אחסון אצלן את נתונים לכלי, השרת האזרוי הוא שמספק את ממשק המשתמש
2. ממשק משתמש - בחרתי לבנות אתר שיספק ממשק משתמש. בהתחלה חשבתי להשתמש באפליקציה על שולחן העבודה, אך לאור יש יתרונות שיותר מתאימים לפרויקט מסווג

זה(לדוגמה, ניתן לגשת ממכשירים שונים ונitin לגשת בקלות ללא הורדה). ממשק המשמש בנייתם באמצעות הספרייה flask בפייתון באמצעות דפי html ו-css.

3. בסיס נתונים - מחסן נתונים שבהם השירות האזורי משתמש. בין היתר, הנתונים נשמרים בסיס הנתונים הם:

- פרטיים על קבוצי המשתמש(כגון גודל הקבוצה, שם הקבוצה וכו').
- פרטי משתמש(כגון שם המשתמש, סיסמה, אימייל וכו').
- קשרים בין משתמשים(בקשות חברות, חברות וכו').

4. שירות מרכזי - שירות מרכזי של כל השירותים האזוריים מתחברים אליו. שירות זה אחראי לאוסף כל 5 דוחות את כל השינויים שנעשו בשירותים האזוריים בזמן הרה וולנסן את שאר השירותים האזוריים(לדוגמה, במידה ומעלים לשירות אזורי קובץ, השירות האזורי שולח זאת לשירות המרכזי שדווג לשולח זאת לכל שאר השירותים האזוריים שמעודכנים זאת אצלם). שירות זה פוטר בעיות עיקריות:

- אין תלות עכשווי בשירות אזורי אחד(במידה ושירות אזורי קורס, אפשר להתחבר מחדש לשירות אחר ולבצע עליו פעולות שונות).
- הזמינות של המערכת גדלה(ניתן לפרוס שירותי אזוריים במקומות שונים).

כל הרכיבים ייחדי מרכיבים מערכת שמספקת אפשרות גיבוי וניהול קבצים בכמה שפותות העבודה מצד המשתמש ושיטוף קבצים בין משתמשים.

### **פירוט היכולות של המערכת:**

היכולות העיקריות של המערכת הן:

- התחברות והרשמה למערכת.
- הוספת קובץ לשרת.
- הורדת קובץ מהשרת למכשיר שמננו מחוברים למערכת.
- מחיקת קובץ מהשרת.
- שינוי הרשאות גישה לקובץ(גישה רק למשתמש, גישה לחברים או גישה ציבורית).
- הוספת חבר.
- מחיקת חבר.

### **פירוט הבדיקות שמתוכננות לפרויקט:**

מספר הבדיקה	שם הבדיקה	מה הבדיקה	איך מתכוונים לבדוק
1	התחברות והרשמה למערכת	האם ניתן להתחבר ולהירשם למערכת	ההרשמה למערכת ולאחר מכן נסיוון התחברות

<b>ביצוע הפעולות הבאות - העלאת קובץ, מיחיקת קובץ, שינוי הרשות קובץ, הורדת קובץ, שליחת בקשה לחברות, אישור בקשה וצפיה בקבצים של חבר</b>	האם המערכת מסוגלת לטפל בפעולות הבסיסיות של לקוח אחד	<b>טיפול בלקוח אחד</b>	<b>2</b>
<b>ביצוע אותן פעולה על גבי מחשב שונה מאשר הרשות רץ על אותו מכשיר</b>	האם המערכת מסוגלת לטפל בלקוח שאינו מחובר על אותו מכשיר	<b>התחברות ממפעיל חיצוני</b>	<b>3</b>
<b>לאחר ביצוע הפעולות, אסגור את הרשות ואבדוק אם הקבצים הרלוונטיים + הפעולות נשמרו</b>	האם השינויים שנעשו נשמרים במדד הנתונים ומערכת הקבצים	<b>טיפול במסד הנתונים ומערכת הקבצים</b>	<b>4</b>
<b>הסנהר באמצעות wireshark תüberort המערכת</b>	האם המערכת מסוגלת להציג את התקשרות	<b>הצפנה המערכת</b>	<b>5</b>
<b>אנסה להיכנס</b>	<b>האם המערכת</b>	<b>ניסיונות</b>	<b>6</b>

<b>למערכת מספר פעמים ברציפות ואראה אם ישנה חסימה לאחר מספר נסיגות מסוים</b>	<b>מסוגלת להתמודד עם brute force</b>	<b>התחברות מתמשכים</b>	
<b>אריך שני שירותים אזוריים משנה מכשירים שונים, בוצע שינוי באחד מהם ואראה אם הוא מתעדכן בשני</b>	<b>האם המערכת מצליחה לסנכרן שינויים בין שניים אזוריים במספר שירותי אזוריים הפועלים בו זמןית</b>	<b>סנכרון שינוי בין שירותים אזוריים</b>	<b>7</b>
<b>באמצעות כלי sqlmap בשם sql injection ואריך שאילתות sql זדוניות ואראה אם המערכת מוגנת מכך</b>	<b>האם המערכת מוגנת מ-sql injection</b>	<b>sql injection</b>	<b>8</b>
<b>אריך את השירותים האזוריים ואכבה את השרת המרכזית, ולאחר מכן מבצע פעולה על מנת לראות אם השרת האזורי מתקף</b>	<b>האם השירותים האזוריים מסוגלים לתפקיד במצב שבו השרת המרכזית קורס</b>	<b>קritishe של השירות המרכזי</b>	<b>9</b>

<p><b>ביצוע מספר שינויים בשרת אזרוי, ולאחר זמן מה הפעלה של שרת אזרוי חדש. אראה אם השינויים שהתרחשו ברשת האזרוי הוטמעו גם ברשת החדש</b></p>	<p><b>האם שרת אזרוי שהתחבר באייחור/אחרי קריסה מסוגל להשלים את השינויים שהתרחשו בזמן זה</b></p>	<p><b>התחברות מאוחרת של שרת אזרוי</b></p>	<b>10</b>
--	--	---	-----------

## תכנון וניהול לוח זמנים:

תחילה התחלתי ממסמכים שונים המאפיינים את הפרויקט, דנים בחלוקת מן הבחריות שבנה ומסבירים בחירות שונות בארכיטקטורת הפרויקט.

המסמכים שבניתי הם:

- מסמך ייזום
- מסמך אפיון
- מסמך ניתוח ועיצוב

לאחר מסמכים אלו, התחלתי לעבוד על הפרויקט עצמו. בתחילת העבודה, השקעת זמן רב במחשבה על מבנה הפרויקט עצמו. חשבתי בעיקר על מספר דרכים שונות למשתמש את העבודה עם מערכת הקבצים עצמה(עליה ארחיב בהמשך התקיק). לאחר מכן, התחלתי עם הקוד של הפרויקט. תחילת התחלתי בפיתוח הלוגיקה של השירותים האזרויים, משם עברתי למשחק המשתמש וסימתי בשורת המרכז וגביו בין השירותים. בתחילת הפרויקט, תכננתי להקנות פחות זמן בפועל ממה שהקצתנו לבניית המשחק הגרפי, דבר שהוביל ל渴צת פחות זמן לעובוד על המשחק לאחר מכן. חוץ מכך, אני מרגש שפעלת יחסית לצורה זהה ללוח הזמנים שתכננתי בהתאם ובסיס הכל עמדתי בו לאורך המטרות הגדולות. אבני הדרך שהצבתי לעצמי במהלך הפרויקט הן:

- בניית הלוגיקה המרכזית של השירות(הוספת קבצים, מחיקתם, פעולות שונות הקשורות לחבריהם ועוד).
- בניית המשחק הגרפי(אתר ועיצוב דפי `html` השונים בפרויקט).
- בניית הלוגיקה של הסyncron בין השירותים האזרויים והשרת המרכזי.
- הוספה אבטחה במקומות הדרישים, תיקון באגים קטנים וליתוש עיצוב הדפים.

- בנוסף לכך, היו 2 מטרות שהצבתי לעצמי אף לבסוף אני ויתרתי עלייהן (שתיהן הרגישו לי לא לגמר):  
 נחוצות ועקב אילוצי זמן ויתרתי עלייהן):
- אפשרות של שילוחת הודעות בין משתמשים
  - אפשרות לשמר כמה גרסאות במקביל של אותו קובץ

### **ניהול הסיכונים בפרויקט:**

הסיכונים העיקריים שחויבתי עליהם במהלך הפרויקט הם:

הסיכון	פירוט הסיכון	תיאור דרכי להטמודד עם הסיכון	מה בוצע בפועל
מערכת לא מוגנת	מערכת בעל חולשות רבות, פגיעה למתקפות שונות	יישום אמצעי אבטחה נגד מתקפות ספציפיות, שימוש בהצפנה לפROYIKT	שימוש ב- <code>sis</code> להצפנה בפרויקט ויישום הגנות נגד מתקפות ספציפיות (ירחב בהמשך)
מערכת לא יציבה	באגים רבים במערכת, יתכוון ל垦יסת המערכת	תכונן נכון של הקוד ושימוש באמצעות <code>try &amp; except</code> תכונן ראשוני של הקוד במידת הצורך שינוי הקוד	שימוש באמצעות <code>try &amp; except</code> , <code>try</code> & <code>except</code> תכונן ראשוני של הקוד במידת הצורך שינוי הקוד
בעיה בסyncron בין שרתים	אי יכולת לsnsron שינויים בין שרתים שונים במערכת	בנייה מנגןון snsron יעיל ויציב	בנייה מנגןון snsron יעיל ויציב

הערכתני את הזמן שלי בצורה לא מספיק טובہ, משימות שלא הקדשתי להן זמן רב בהתחלתו לקחו לי הרבה יותר זמן משחשת ובפועל היו תקופות שהתקשתי להתמיד בפרויקט עקב סיבות שונות	ניהול הזמן בצורה נכון ופriseת העבודה בצורה חכמה	עיכוב בהגשת הפרויקט	אי עמידה בלוחות הזמן
---	---	------------------------	-------------------------

## טיואר תחום הידע:

### צד לקוחות:

#### 1. רישום משתמשים:

- מהות היכולת - ייצרת חשבון חדש במערכת
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:

  - תצוגת טופס הרשמה(שם משתמש, אימייל, סיסמה)
  - כפתור "הרשמה"
  - בדיקת תקינות הנתונים - שדות לא ריקים, כתובות אימייל תקינה, סיסמה עמידה בתנאים
  - הצפנה - hashing של הסיסמה, העברת הנתונים בצורה מאובטחת(שימוש ב-zts)
  - שליחה אל השרת - בקשה post

- קבלת הנתונים בשרת
  - בדיקת תקינות בשרת - ייחודיות בפרטים הדורשים לכיר
  - כתיבת הנתונים במסד הנתונים - יצירת user חדש
  - הצגת הודעה הצלחה/שגיאה מתאימה
  - סנכרון האירוע - הוספו לתור השומר את האירוע עד לבקשת הסנכרון של השירות המרכזי
  - הפניה לעמוד ההתחברות
- אובייקטים נוחצים - מסך הרשמה, מסד נתונים, פונקציית hash לשימוש, תקשורת בין השירות ללקוח, תעודה אס ופתח ציבור, חיבור לשרת המרכזי.

## **2. התחברות ואימות משתמש:**

- מהות היכולת - התחברות של משתמש קיים וקבלת session מאובטח.
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
- מסך התחברות(שדות להכניס אימייל וסיסמה)
  - כפתור "התחברות"
  - בדיקת תקינות הנתונים - שדות לא ריקים, כתובות אימייל תקינה/קיים במערכת, סיסמה התואמת את האימייל
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחה אל השירות - בקשה post
  - קבלת הנתונים בשרת
  - בדיקת תקינות בשרת - השוואת ערך ה-hash של הסיסמה הנשלחת לסיסמה הקיימת במסד הנתונים
  - יצירת session token מותאים במידה והפרטים תקינים
  - הפניה לעמוד הראשי
- אובייקטים נוחצים - מסך התחברות, מסד נתונים, פונקציית hash לשימוש, תקשורת בין השירות ללקוח, תעודה אס ופתח ציבור.

## **3. צפיה בקבצים:**

- מהות היכולת - צפיה בקבץ המשתמש בדף הבית
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
- מסך בית
  - שליפת הקבצים השיעיכים למשתמש ממasd הנתונים
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - הצגת הנתונים על המסך
- אובייקטים נוחצים - עמוד בית, מסד נתונים, תקשורת בין השירות ללקוח.

## **4. העלאת קבצים:**

-מהות היכולת - העלה מאובטחת של קובץ לשרת.

-אוסף היכולות/פעולות הנדרשות לימוש היכולת:

- עמוד בית
  - כפתור "בחר קובץ"
  - בדיקת תקינות קובץ - גודלו אינם גדול מדי, הסימות של הקובץ מורשתית
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחת הקובץ לשרת
  - קבלת הנתונים בשרת
  - ייצירת שם ייחודי
  - שמירת הקובץ במערכת הקבצים
  - הוספה למסד הנתונים - הוספה קובץ חדש, עדכון שדות בטבלת המשתמשים
  - הצגת הודעת הצלחה/שגיאה מתאימה
  - סנכרון האירוע - הוסpto לטור השומר את האירוע עד לבקשת הסנכרון של השרת המרכז'
  - הצגת התוצאה במסך הבית - הוספה הקובץ לרשימה הקבצים
- אובייקטים נוחצים - מסד נתונים, מסך בית, תקשורת בין השרת ללקוח, טור השומר את השינויים, חיבור לשרת המרכז'.

## **5. הורדת קבצים:**

-מהות היכולת - הורדת מאובטחת של קבצים מהשרת למכשיר הלקוח.

-אוסף היכולות/פעולות הנדרשות לימוש היכולת:

- עמוד בית
  - כפתור "הורדה"
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - הורדת הקובץ למכשיר הלקוח
  - הצגת הודעת הצלחה/שגיאה מתאימה
- אובייקטים נוחצים - תקשורת בין השרת ללקוח, מסך בית.

## **6. מחיקת קובץ:**

-מהות היכולת - מחיקת קובץ מהשרת

-אוסף היכולות/פעולות הנדרשות לימוש היכולת:

- עמוד בית
- כפתור "מחיקה"
- שליחת הקובץ הרצוי למחיקה לשרת
- מחיקת הקובץ ממערכת הקבצים
- מחיקת הקובץ ממסד הנתונים
- סנכרון האירוע - הוסpto לטור השומר את האירוע עד לבקשת הסנכרון של השרת המרכז'
- הצגת הודעת הצלחה/שגיאה מתאימה

- הצגת התוצאה במסך הבית - מחיקת הקובץ מרשימת הקבצים
- אובייקטים נוחצים - תקשורת בין השרת ללקוח, מסך בית, תור השומר את השינויים, חיבור בין השרת לשרת המרכזי, מסד נתונים.

## **7. שינוי הרשאות של קובץ:**

- מהות היכולת - שינוי הרשות הגישה של משתמשים אחרים לקובץ של המשתמש ('פרט', 'חברים בלבד', 'ציבורי').
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
  - עמוד בית
  - כפתור "שינוי הרשות"
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחת השינוי לשרת
  - קבלת הנתונים בשרת
  - עדכון מסד הנתונים - שינוי הרשות של הקובץ המתאים לכך
  - הצגת הודעת הצלחה/שגיאה מתאימה
  - הצגת התוצאה על המסך - שינוי הרשות בראשית הקבצים
  - סנכרון האירוע - הוספו לתור השומר את האירוע עד לבקשת הסנכרון של השרת המרכזי
- אובייקטים נוחצים - תקשורת בין השרת ללקוח, תקשורת בין השרת לשרת המרכזי, מסך בית, מסד נתונים, תור השומר את השינויים.

## **8. שליחת בקשה חברות:**

- מהות היכולת - שליחת בקשה חברות למשתמש אחר הקים במערכת.
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
  - עמוד שליחת בקשות חברות
  - כפתור "שליחת בקשה"
  - בדיקת תקינות הנתונים - הקלט אינו ריק
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחת הנתונים לשרת
  - קבלת הנתונים לשרת
  - בדיקת תקינות הנתונים - המשתמש קים, המשתמש אינו חבר של המשתמש השני, לא קיים בקשה המחייב לאישור
  - עדכון מסד הנתונים - הוספה לטבלה בקשות החברות
  - סנכרון האירוע - הוספו לתור השומר את האירוע עד לבקשת הסנכרון של השרת המרכזי
  - הצגת הודעת הצלחה/שגיאה מתאימה
- אובייקטים נוחצים - מסך בקשות חברות, תקשורת בין השרת ללקוח, תור השומר את השינויים, מסד נתונים, תקשורת בין השרת לשרת המרכזי.

## **9. הצגת בקשות חברות:**

- מהות היכולת - הצגות בקשות חברות שנשלחו למשתמש
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
  - עמוד בקשות חברות
  - שליפת הביקשות השויות למשתמש מסוד הנתונים
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחת הנתונים ללקוח
  - הצגת הנתונים במסך הלקוח
- אובייקטים נוחצים - מסך בקשות חברות, תקשורת בין השירות ללקוח, מסד נתונים.

## **10. אישור/דחית בקשה חברות:**

- מהות היכולת - אישור/דחיה של בקשת חברות הנשלחה למשתמש.
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
  - עמוד בקשות חברות
  - הצגת בקשות חברות
  - כפטור "אישור ו"דחיה"
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחת התשובה לשרת
  - קבלת הנתונים בשרת
  - עדכון מסד הנתונים - שינוי סטטוס הבקשה(ובעת הצורך להוריד אותה)
  - במידה והבקשה אושרה - עדכון מסד הנתונים - הוספה לטבלת החברויות הקיימות
  - סנכרון האירוע - הוספתו לתוך השומר את האירוע עד לביקשת הסנכרון של השירות המרכזי
  - הצגת הודעה הצלחה/שגיאה מתאימה
  - מחיקת הבקשה מסך הבקשות
- אובייקטים נוחצים - מסך בקשות חברות, תקשורת בין השירות ללקוח, מסד נתונים, תור השומר את השינויים, תקשורת בין השירות לשרת המרכז.

## **11. צפיה ברשימת החברים:**

- מהות היכולת - צפיה ברשימת החברים של המשתמש.
- אוסף היכולות/פעולות הנדרשות למימוש היכולת:
  - עמוד חברים
  - שליפת החברים של המשתמש מסוד הנתונים
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - הצגת החברים על המסך
- אובייקטים נוחצים - מסך חברים, תקשורת בין השירות ללקוח, מסד נתונים.

## 12. צפיה בקבצים של חבר:

-מהות היכולה - צפיה בקבצים שהחבר שול המשמש אישר גישה אליום (מצב 'חברים בלבד' או 'ציבור').

-אוסף היכولات/פעולות הנדרשות למימוש היכולה:

- כפטור "צפה בקבצים"
- עמוד קבצים של חבר
- שליפת הקבצים שמאושרת הגישה אליהם ממסד הנתונים
- הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
- הצגת הקבצים על המסך

-אובייקטים נוחוצים - מסך קבצים של חבר, תקשורת בין השירות ללקוח, מסד נתונים.

## 13. הסרת חבר:

-מהות היכולה - הסרת משתמש מרשימה החברים של המשמש הקיימים.

-אוסף היכولات/פעולות הנדרשות למימוש היכולה:

- עמוד חברים
  - כפטור "הסרת חבר"
  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-tls)
  - שליחת הנתונים לשרת
  - קבלת הנתונים בשרת
  - שינוי במסד הנתונים - הסרת החבר מטבלת החברויות
  - סנכרון האירוע - הוסpto לתוך השומר את האירוע עד לבקשת הסנכרון של השירות המרכזי
  - הצגת הוועדת הצלחה/שגיאה מתאימה
  - מחייקת החבר ממשך החברים
- אובייקטים נוחוצים - מסך חברים, תקשורת בין השירות ללקוח, מסד נתונים, תור השומר את השינויים, תקשורת בין השירות לשרת המרכזי.

## 14. התנטקיות מהמערכת:

-מהות היכולה - התנטקיות מהמערכת וזרה למסך ההתחברות.

-אוסף היכولات/פעולות הנדרשות למימוש היכולה:

- כפטור התנטקיות
- שליחת הנתונים לשרת
- קבלת הנתונים בשרת
- הפניה לעמוד ההתחברות

-אובייקטים נוחוצים - כפטור התנטקיות, עמוד התחברות, תקשורת בין השירות ללקוח.

## צד שרת:

### 15. שליחת שינויים לשרת:

- מהות היכלota - עדכו shinonim שהתרחשו בשרת האזרוי לשרת המרכז.
- אוסף היכלות/פעולות הנדרשות למימוש היכלות:

  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-*Isat*)
  - הוספה הנתונים לאחר כל אירוע לטור - *changes*(מבנה נתונים מסווג טור)
  - קבלת הודעה מהשרת המרכז המცביעה על סyncronizaion הנתונים
  - שליחת האירועים שבטור לשרת המרכז'

- אובייקטים נוחוצים - תקשורת בין השירות לשרת המרכז, מסד נתונים, טור השומר את השינויים, סוקטים

### 16. עדכון שירותי אזרויים בשינוי:

- מהות היכלota - עדכו shinonim שהתרחשו בשרת אזרוי מסוים לשאר השירותים האזרויים.
- אוסף היכלות/פעולות הנדרשות למימוש היכלות:

  - הצפנה - העברת הנתונים בצורה מאובטחת(שימוש ב-*Isat*)
  - הבחנה מייזה שירות אזרוי הגיע עדכון מסוים
  - שליחת העדכון לשאר השירותים האזרויים
  - קבלת הנתונים בשירותים האזרויים
  - עדכון מסד הנתונים בהתאם לסוג השינוי

- אובייקטים נוחוצים - תקשורת בין השירות לשרת המרכז, מסד נתונים, טור השומר את השינויים, סוקטים

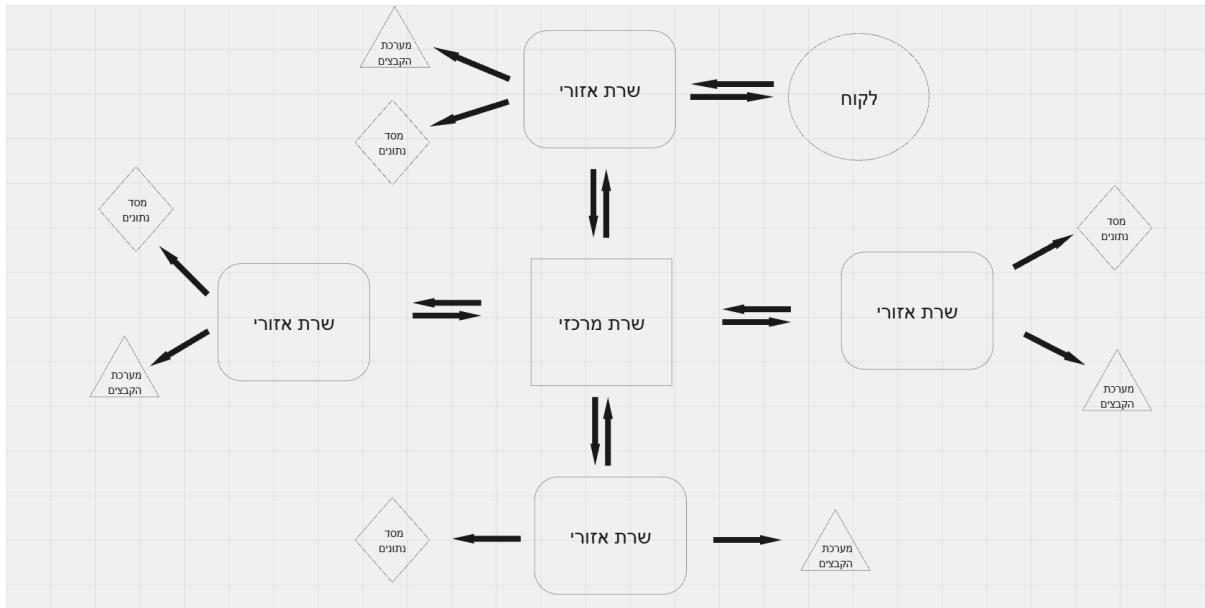
## מבנה/ארQUITקטורה של פרויקט:

### תיאור הארכיטקטורה של המערכת המוצעת:

תיאור החומרה:

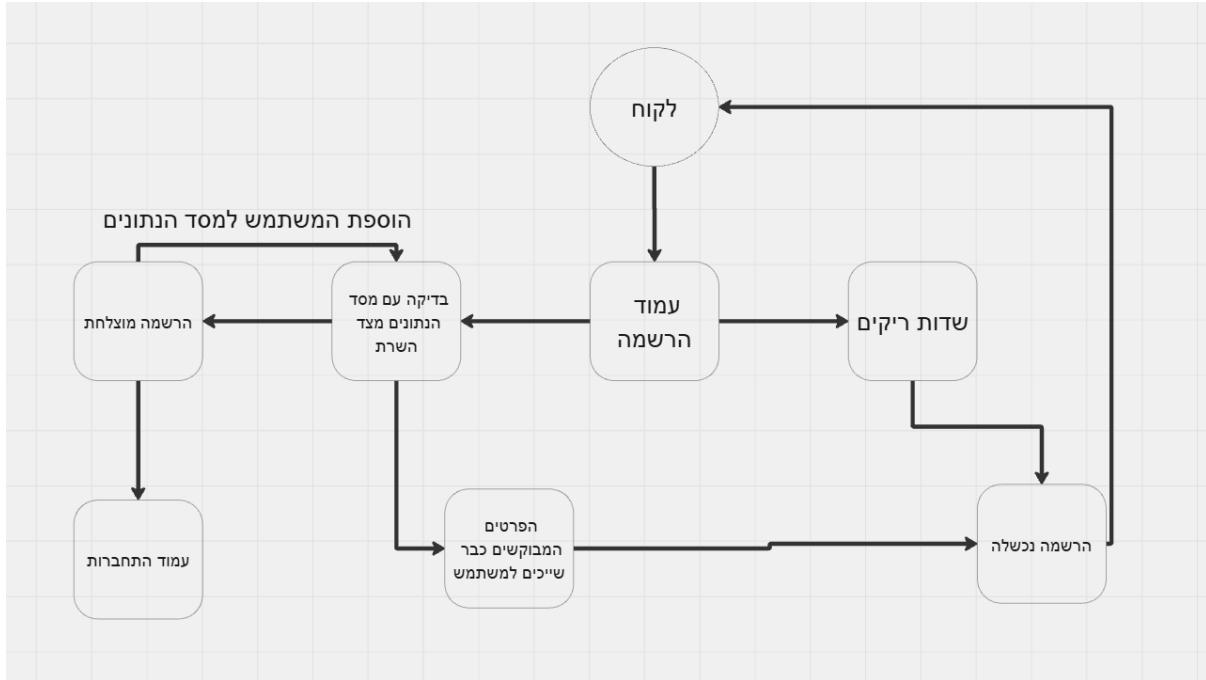
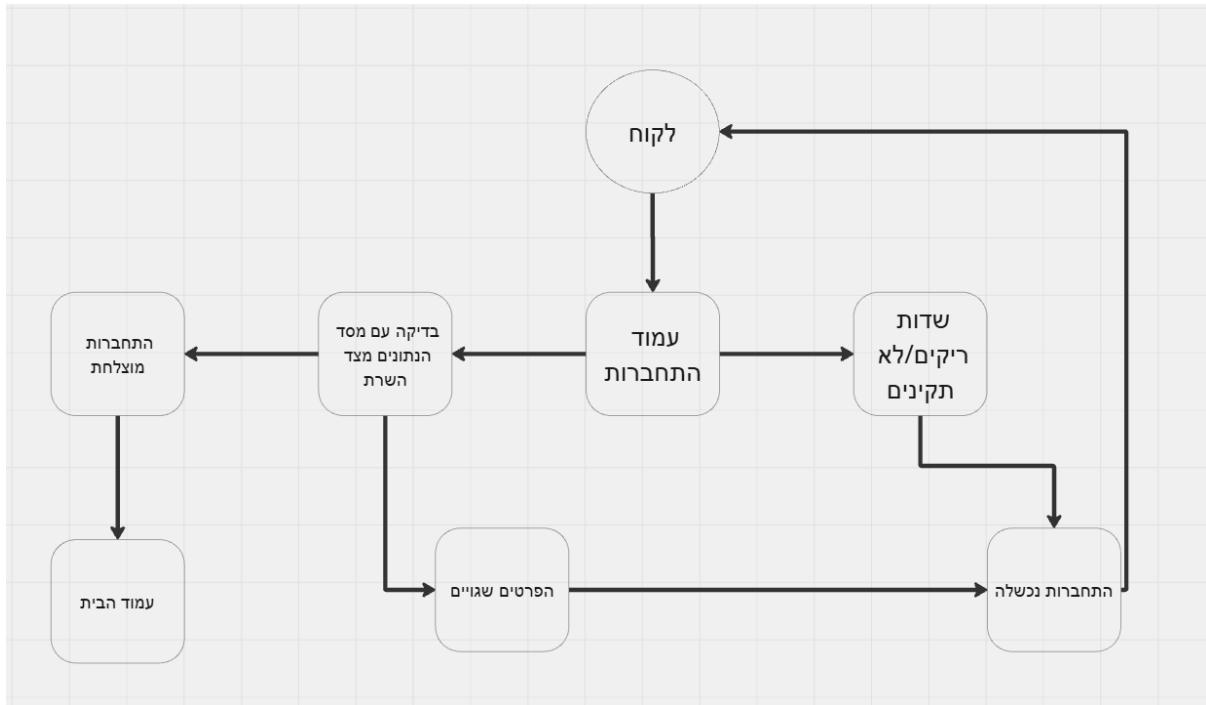
1. מסד נתונים
2. שירות אזרוי - שירות web מבוסס על flask.

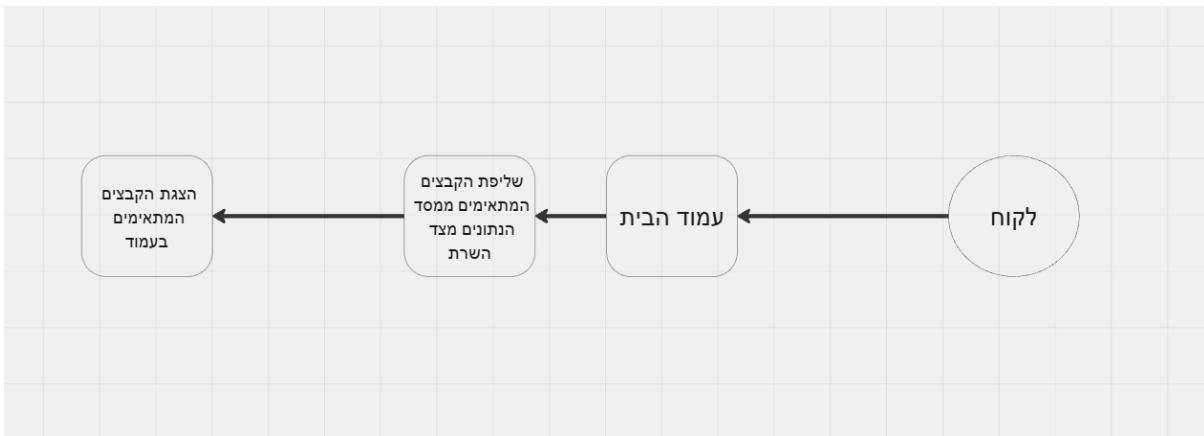
3. שרת מרכזי - שרת מבוסס סוקטים  
 4. תיק'ית הקבצים - אחסון הקבצים במערכת  
 קשרים בין רכיבי החומרה - בין הלקוח לשורת האזרוי יש קשר דו ציווינ'(כל צד מתקשר עם השני),  
 וכך גם קשר דו ציווינ' בין השירות המרכזי לשורות האזרויים. בנוסף לכך, השירותים האזרויים עובדים  
 מול תיקיות הקבצים ומסדר הנתונים.



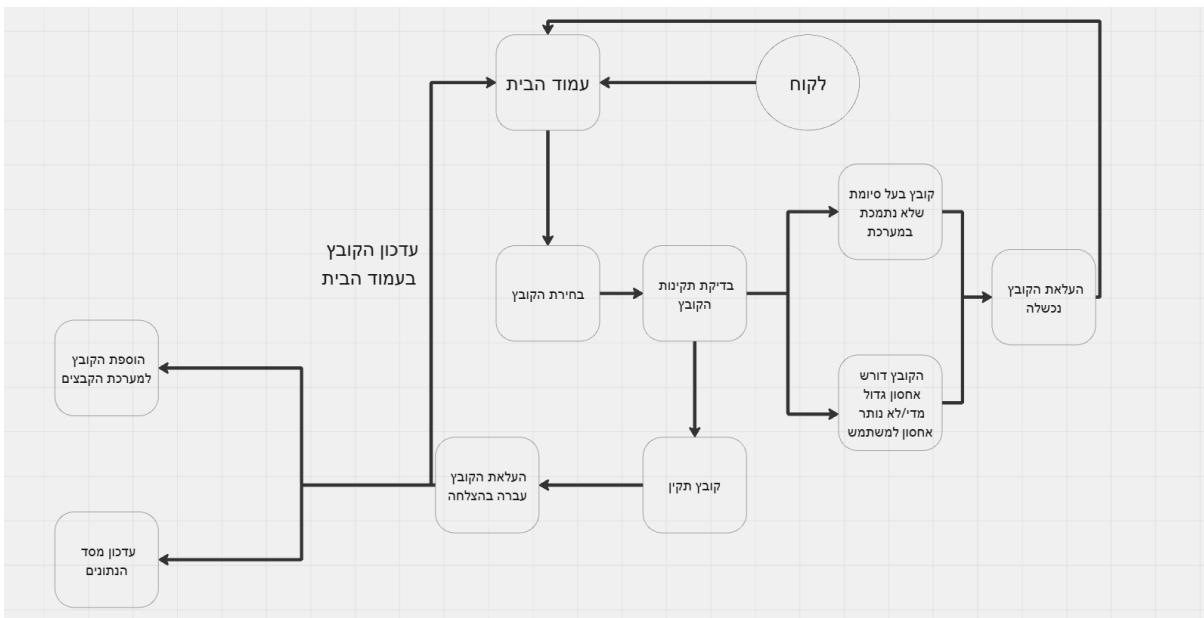
### **תיאור הטכנולוגיה הרלוונטית:**

- ממשק גרפי - שימוש ב-`html`, `css` ו-`js`
- מודול נתונים - `sqlite` ו-`MySQL`
- הצפנה - הצפנה עקב שימוש ב-`TLS`(גרסה זו של הпротокול משתמש ב-`AES`(הצפנה סימטרית) על מנת לשתף את המפתחות ולאחר מכן ב-`RSA`(הצפנה אסימטרית)).
- תקשרות - תקשורת מבוססת סוקטים `tcp`, בנוסף לכך, בין השירות ללקוח ישנה תקשורת `https` מבוססת `TLS` (פרוטוקול `https`)
- מערכת הפעלה - `windows` ו- `python 3.12`

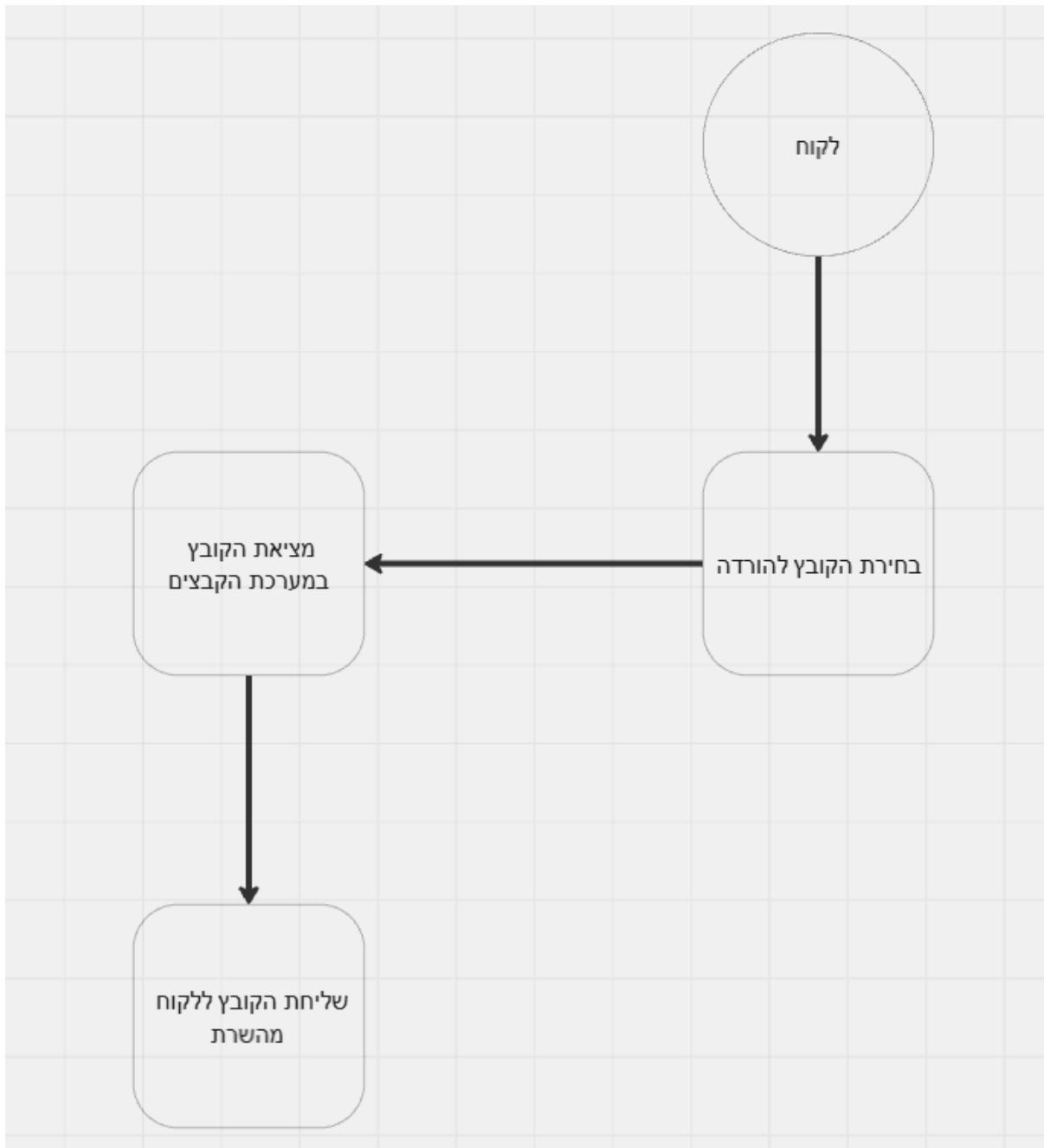
**תיאור זרימת המידע במערכת:****1. הרשמה למערכת****2. התחברות למערכת****3. צפיה בקבצים**



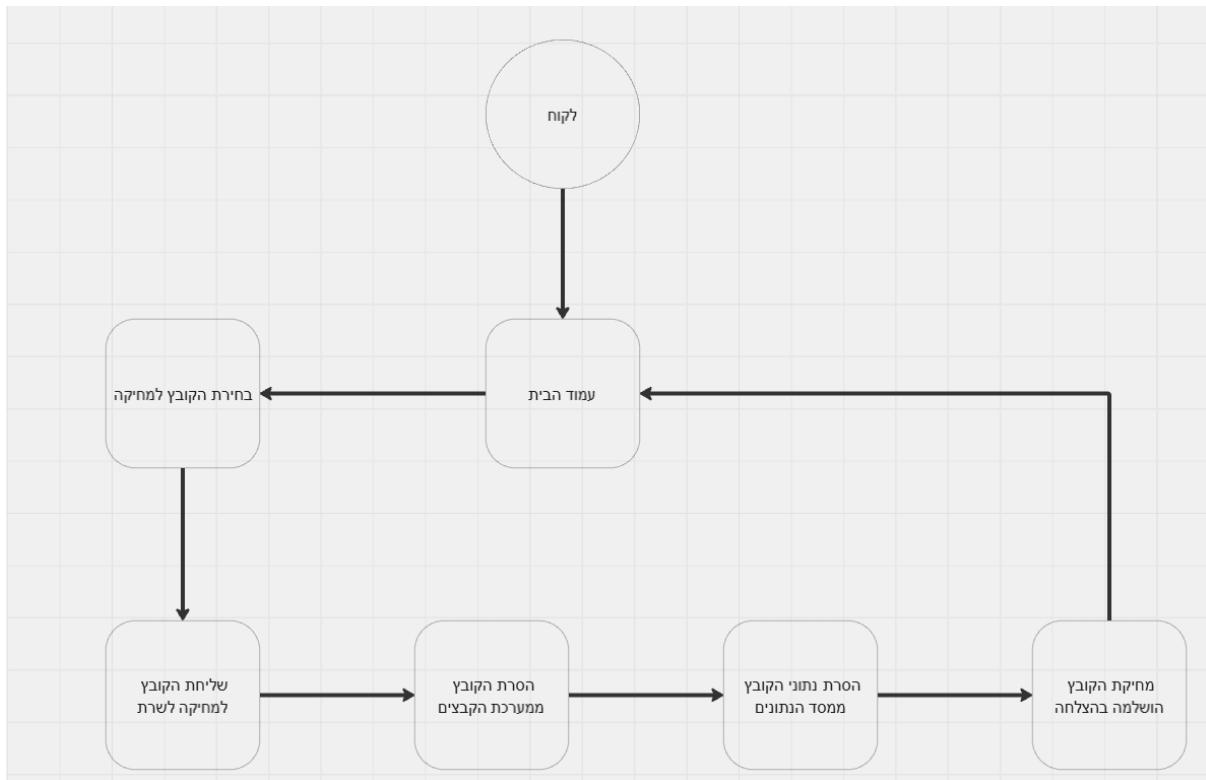
#### 4. העלאת קובץ



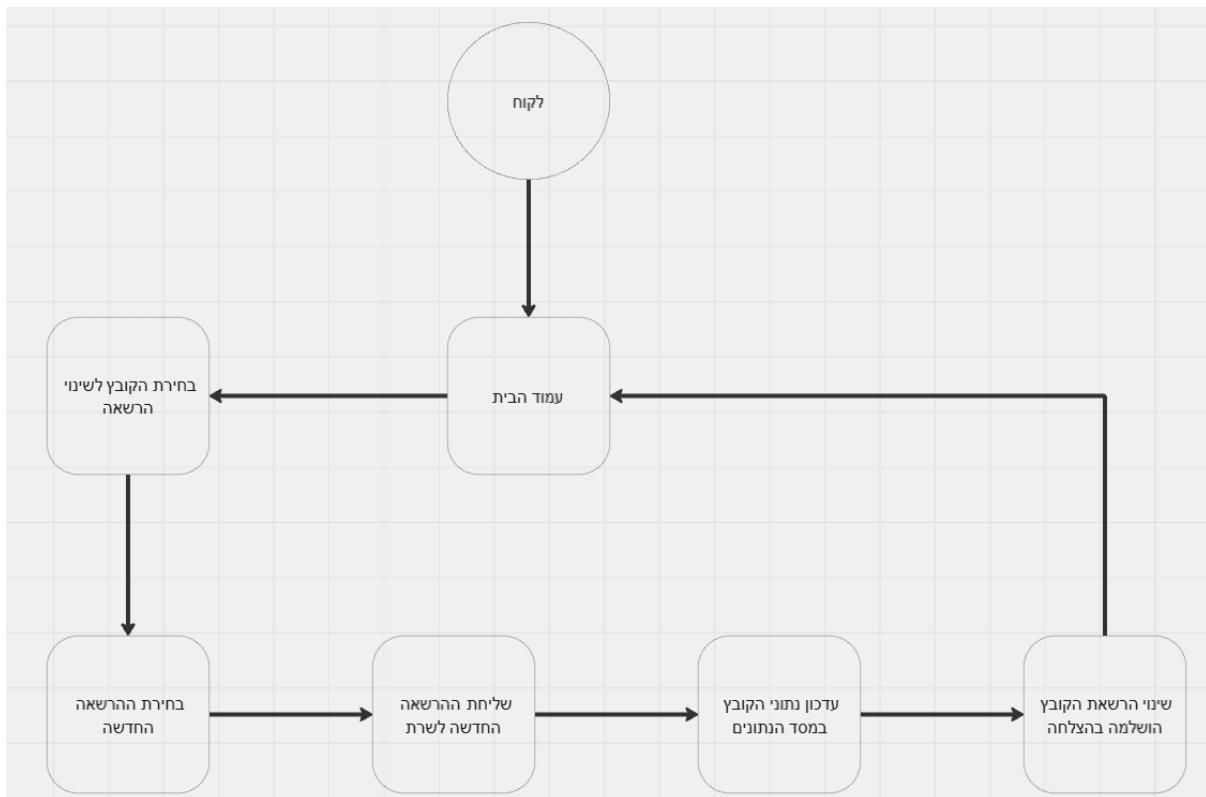
#### 5. הורדת קובץ



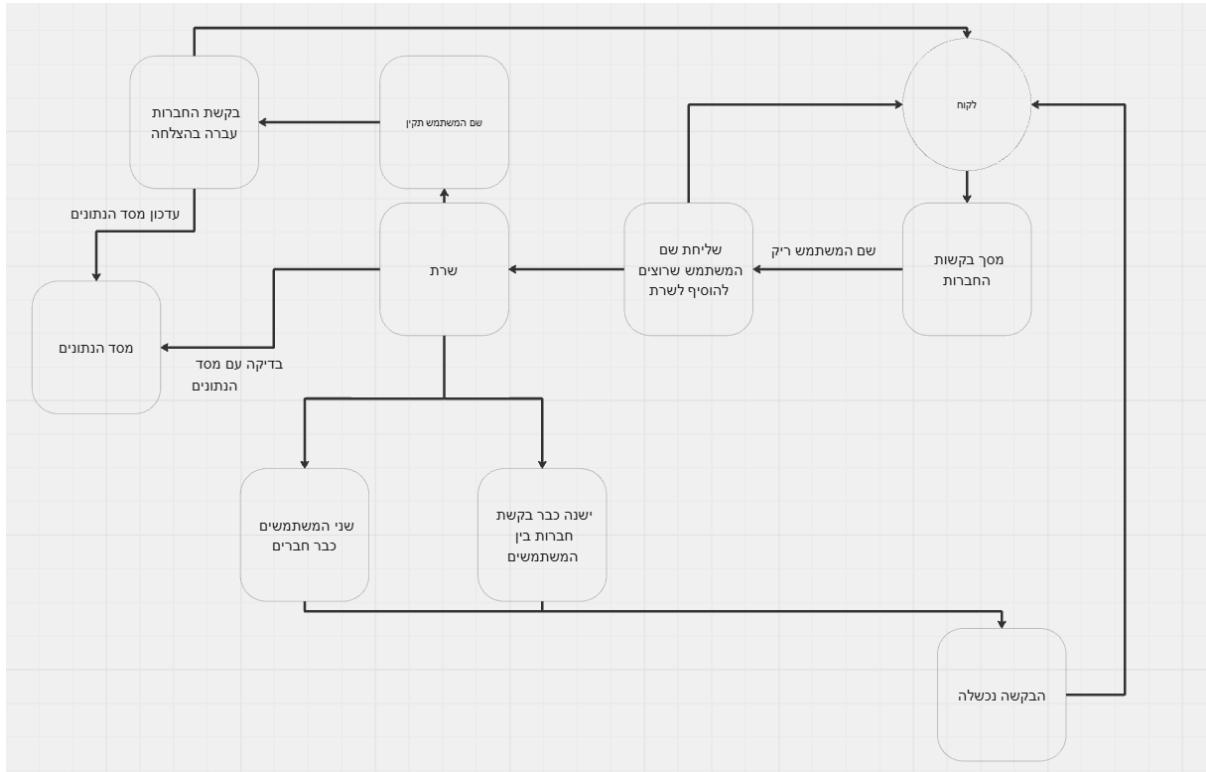
## 6. מחיקת קובץ



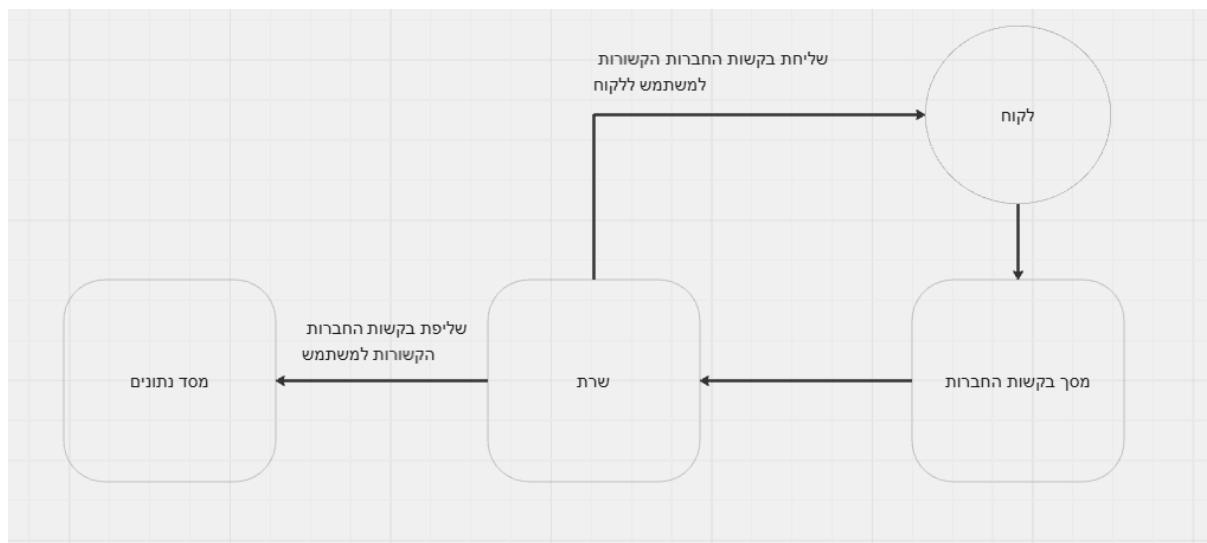
## **7. שינוי הרשאה של קובץ קיים**



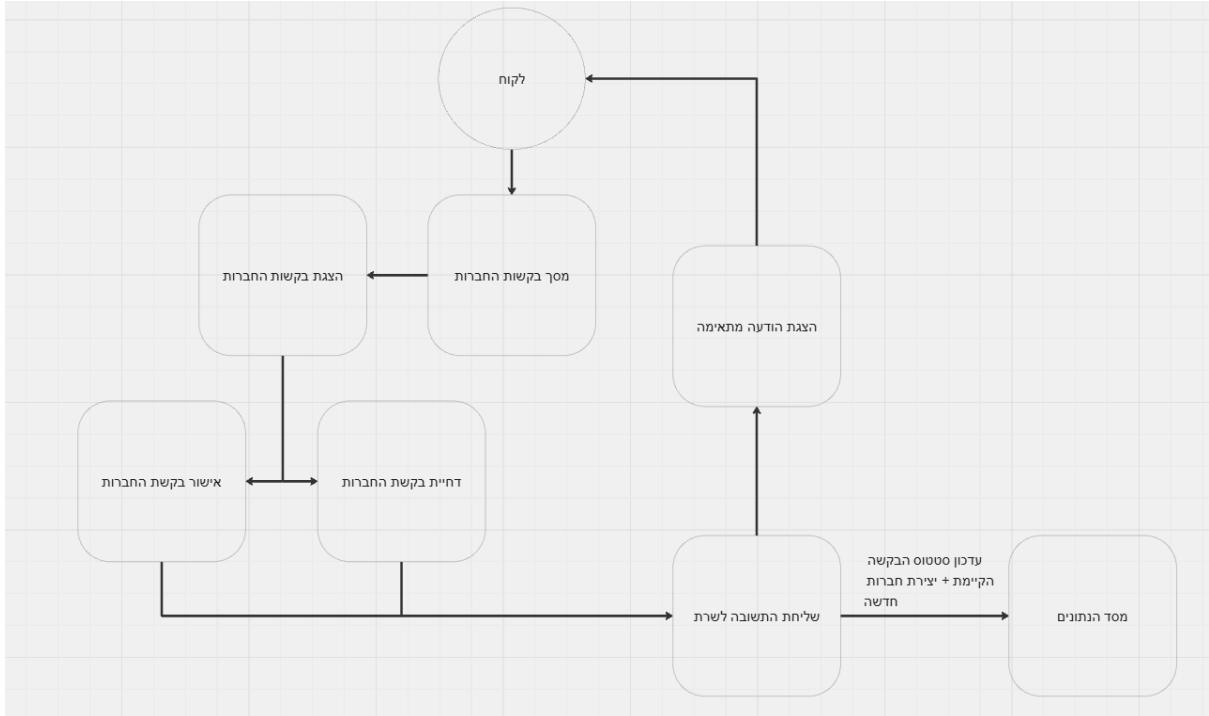
## **8. שליחת בקשה לחברות**



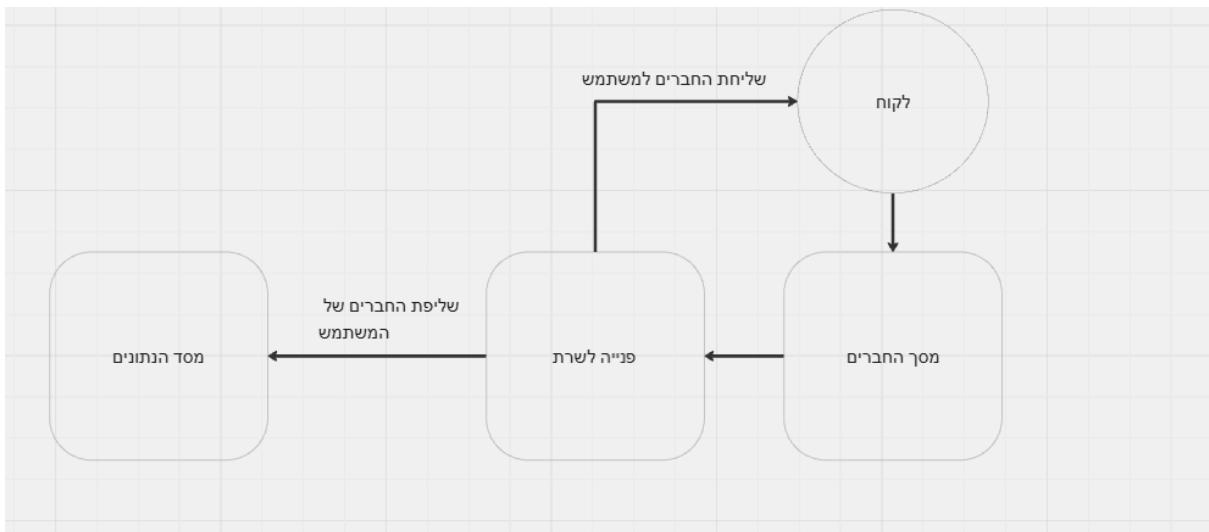
## **9. הצגת בקשות החברות**



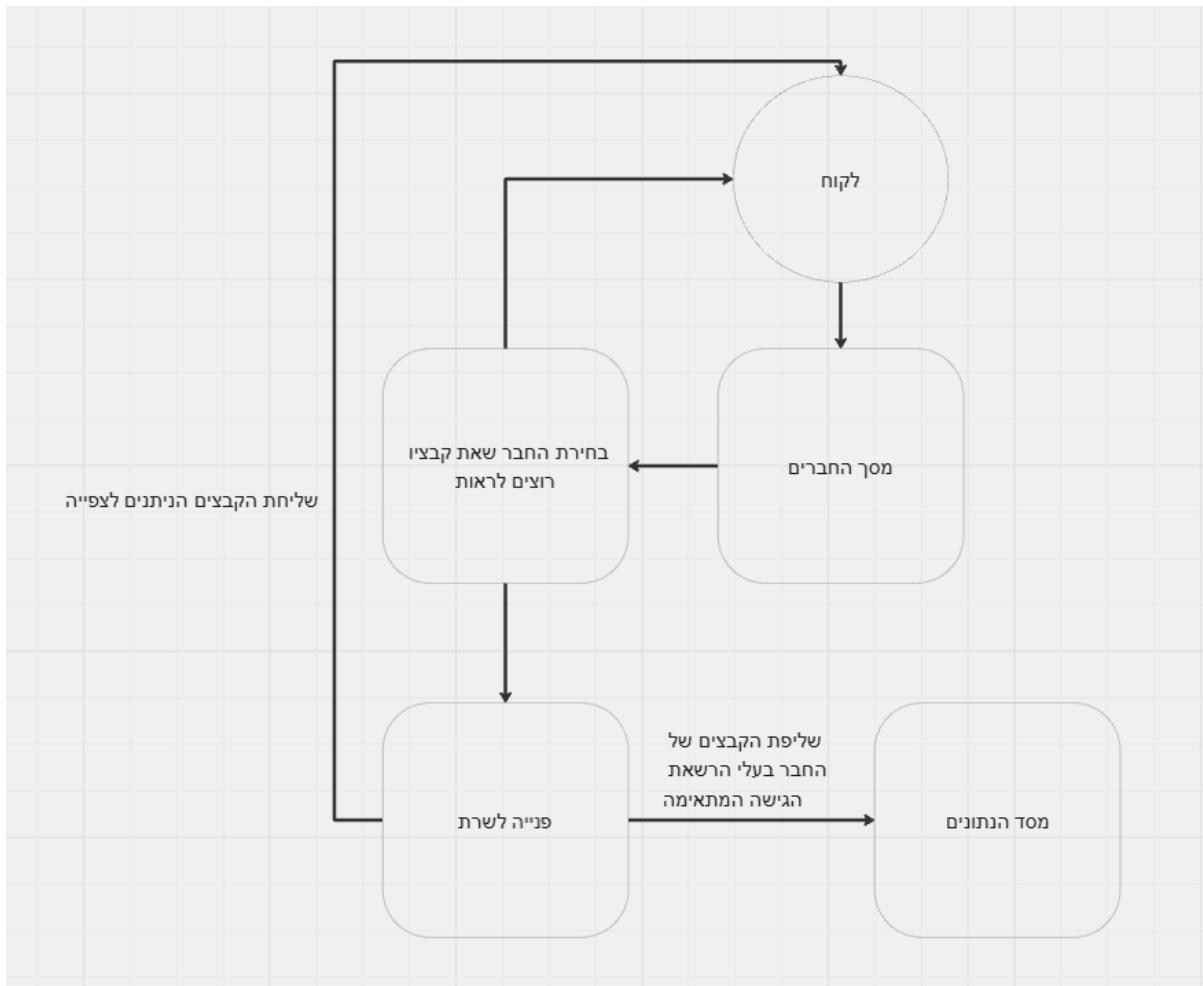
## **10. אישור/דחיית בקשת חברות**



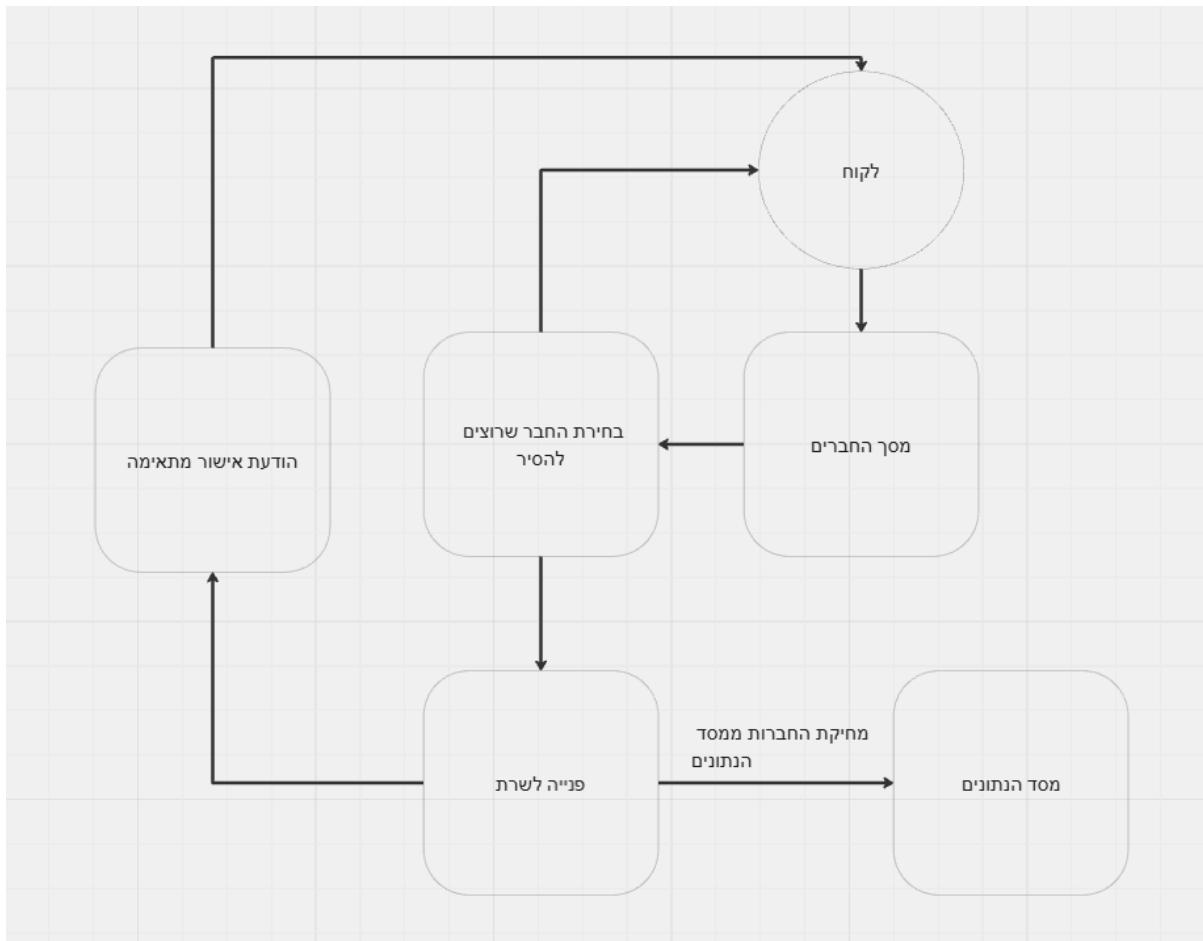
## 11. צפיה ברשימה החברים



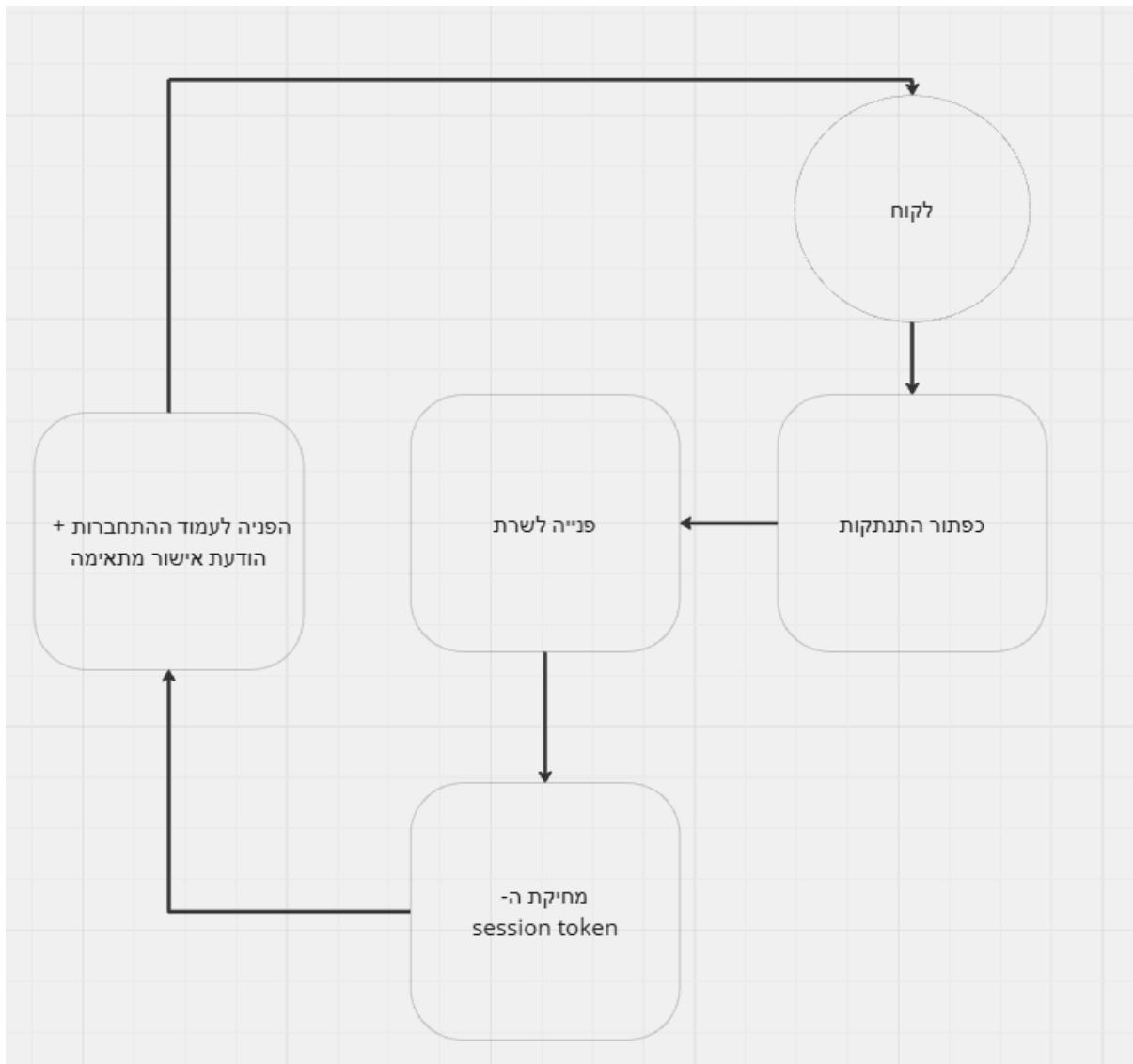
## 12. צפירה בקבצים של חבר



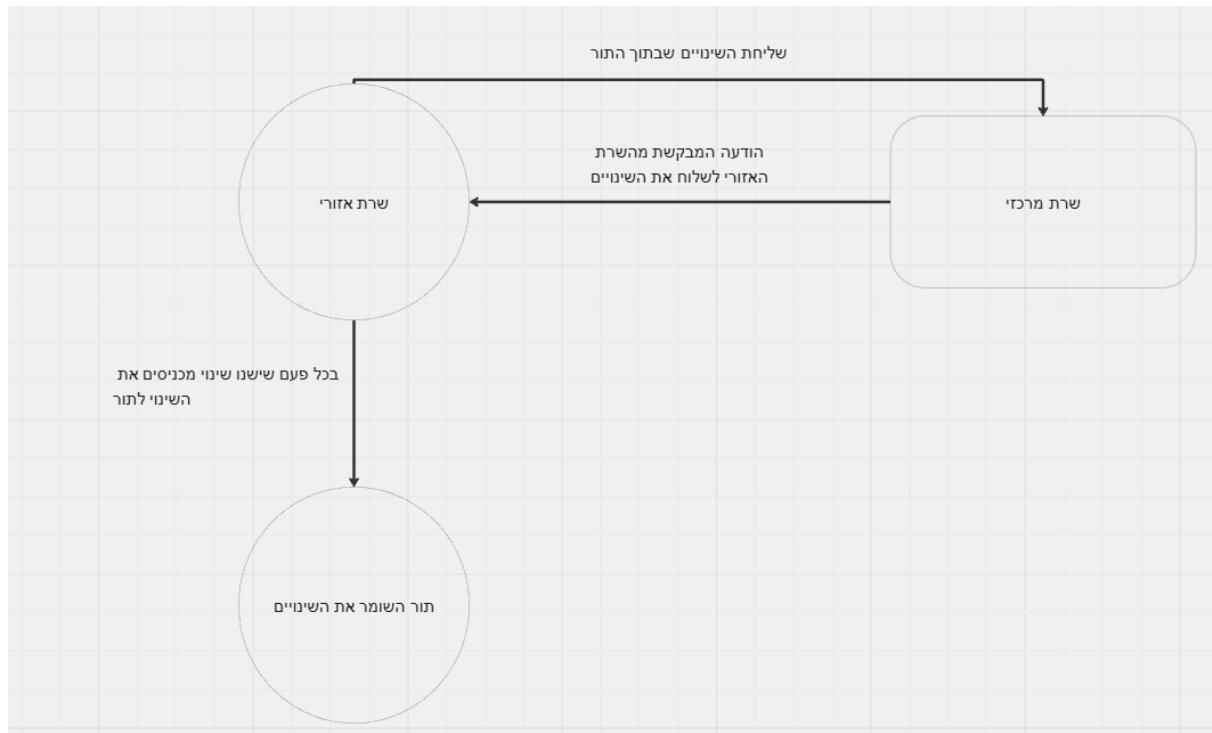
### **13. הסרת חבר**



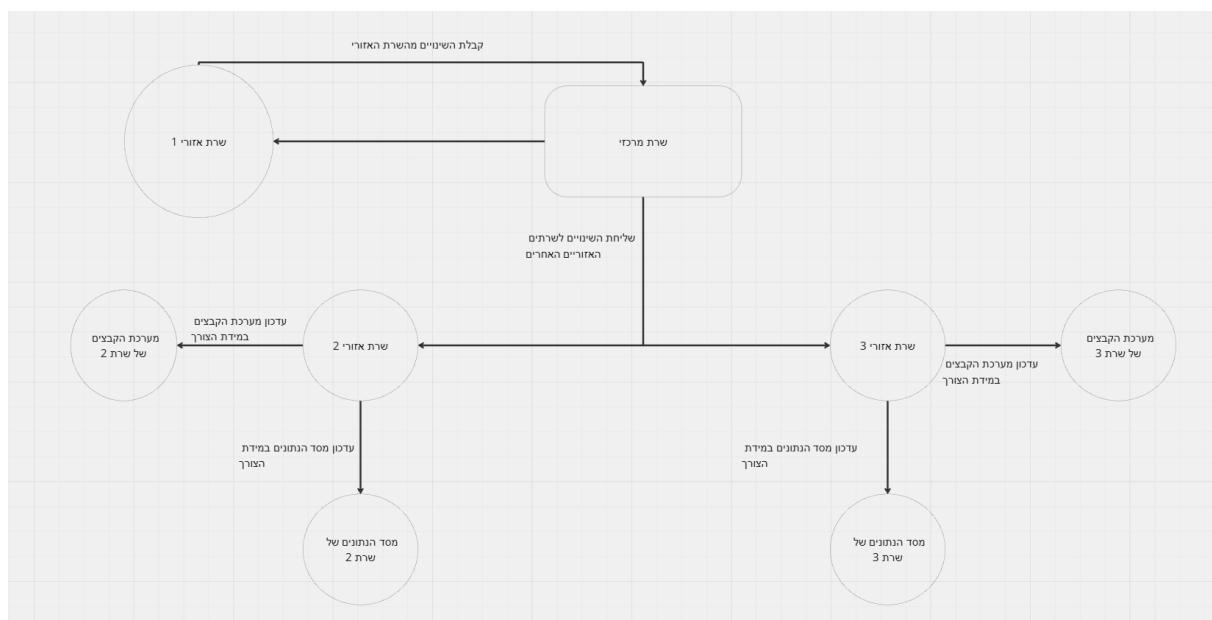
## 14. התנתקות מהמערכת



**צד שרת:**  
**15. שליחת השינויים לשרת**



## 16. עדכון שרתים אזרוריים בשינויי



## **תיאור האלגוריתמים במערכת:**

### 1. מבנה מערכת הקבצים:

הבעיה - הפרויקט כולל עבודה מתמשכת עם מערכת הקבצים של השירותים. בניית מערכת קבצים במצבה הנוכחי בעלת השפעה בביטחון המערכת ומהירות המערכת.

**حلופה 1** - יצירת תיקייה לכל משתמש. לחולופה זו יש מספר יתרונות. לדוגמה, הקבצים של כל משתמש מבודדים בתיקייה נפרדת כך שאם משהו קורה לאחת מן התקיות, לא תהיה השפעה על קבצי שאר המשתמשים. בנוסף לכך, כל עוקב אחר יכול להשתמש(אחר כמות הזכרן שנותר לו לדוגמה) בחולופה זו. לעומת זאת, יש לפתורן זה גם מספר חסכנות.

פתרון זה עשוי להפוך את מערכת הקבצים למסורבלת מאוד ומלאה בתיקיות שונות.

**حلופה 2** - שמירת כל הקבצים בתיקייה אחת כאשר לכל קובץ נתונים שם ייחודי באמצעות פון (מזהה ייחודי שדואג להבטיח שלכל אובייקט יהיה מזהה ששייך רק לו). היתרון העיקרי של חולופה זו היא פשוטות השימוש. בנגדוד לחולופה השנייה, כמעט ולא צריך לנוט במערכת הקבצים במהלך ריצת הפרויקט. החסרון העיקרי של חולופה זו היא שהרבה יותר קשה לעוקב אחר משתמש ספציפי ולבצע פעולות הקשורות אליו(לדוגמה, במידה ורוצים למחוק משתמש מהמערכת, צריך לעבור על כל קובץ על מנת למחוק פשוט למחוק תיקייה מסוימת).

תחליה חשבי שחלופה הראשונה היא החלופה העדיפה לפרויקט, עקב היתרונות שלה בبنיה מערכת בקנה מידה גדול יותר. אך עקב הפשטות של החלופה השנייה והגמישות שבפתרון זה(במידה ומתחילה למשם פתרון זה ורוצים להחליף פתרון הרבה יותר פשוט מאשר הפתרון השני).

### 2. תלות בשרת אחד:

הבעיה - עקב כך שהשרת בפרויקט צריך לשמור את הנתונים ואת הקבצים של כל המשתמשים, נוצרת תלות בשרת זה. קריסה של השירות/עומס של קוחות רבים על אותו שרת עלולה לפגוע ביצועי המערכת ולייצור תלות בשרת זה.

**حلופה 1** - בניית שרת מרכזי האחראי על סנכרון בין הרבה שירותים אזוריים(סוג של שימוש ב-topology). חולופה זו פותרת את הבעיה בכך שבמידה ושרת מסוים קורט, עדין ניתן לגשת למערכת ולהשתמש בשירותה. היתרונות המרכזים של חולופה זו היא שלஸפר השירותים האזוריים אין כל כך השפעה על ביצועי המערכת(כל עוד השירות המרכזי יכול לטפל בכמות השירותים בו זמןנית) וכך שניתן במידת הצורך לפרוס מספר רב של שירותים אזוריים באזוריים שונים(דבר יכול לשדרג את הזמן). החסכנות המרכזים של חולופה זו היא שנוצרת בעצם תלות בשרת המרכזי כך שאין פועל זמן-

**حلופה 2** - שימוש ב-*mesh topology*(טופולוגיה רשת המורכבת ממספר רב של צמתים המשמשים גם כמקור מידע וגם על מנת להעביר מידע מרשת לשרת). חולופה זו מאפשרת מספר רב של שירותים פרויסים וכך אינה תלות בשרת כלשהו. היתרון המרכזי של חולופה זו היא שאין תלות בשרת מסוים במערכת כך שבמידה ושרת מסוים קורט, המערכת תמשיך לתפקיד אותה דרך בדיק. החסרון העיקרי של חולופה זו הוא שכך יש מספר רב יותר של שירותים, כך מבנה הרשת נהייה מסובך יותר יותר.

החלופה שבחרתי במקרה זה היא החלופה הראשונה. אני חשב שהחסרון המרכזי של החלופה 1 הוא שהוא שnitן להתגבר עליו (לדוגמה, בניית מספר שירותים מרכזיים המתקשרים ביניהם במקרה אחד מן השירותים המרכזיים נופל). בניגוד לחלופה 2, ניתן להתגבר על החסרון של חלופה 1 בצורה יחסית פשוטה.

## תיאור סביבת הפיתוח:

- שפת תכנות - python 3.12
- עיצוב משתמש הגרפי - html, css
- כלים פיתוח - visual studio code
- מסד נתונים - sqlite 3

## תיאור פרוטוקול התקשרות:

פרוטוקול התקשרות שיצרתי מומッシュ בעיקר בסמכרו הפעולות החדשנות בין כל שירות. פרוטוקול התקשרות בניו כך:  
שדה 1: סוג הודעה - מחלקת 1-2 סוגים שונים, הודעה שהשרת המרכזי שלח והודעה שהשרת אזורי שלח. במידה והשרת המרכזי שלח את ההודעה, סוג הודעה הוא:

- שלח (send) - סימן לשרת האזורי לשלוח את כל השינויים שהוא צבר במהלך ריצת השירות. אחרי הודעה זו לא ניתן כלום.
- קבל (receive) - סימן שהודעה זו כוללת את כל השינויים שעל השירות לעדכן. השדה הנוסף שיתקבל בסוג הודעה זו הוא 'events' והוא העדכנים שהשרת צריך לשנות (עליהם ארכחיב בהמשך).

במידה והשרת האזורי שלח לשרת המרכזי עדכון מסוים, סוג הודעה הוא סוג האירוע שעלה השירותים האחרים לעדכן. לכל הודעה יש payload שהוא הנ吐nomים הדורשים לביצוע השינוי. האירועים האפשריים הם:

- העלאת קובץ (file\_upload) - קובץ חדש הועלה לשרת. ה-payload הוא id הקובץ, id המשמש, שם הקובץ המקורי, שמו במערכת הקבצים, תאריך הupload, גודל הקובץ, הרשאה ותוכן הקובץ.
- מחיקת קובץ (file\_delete) - קובץ נמחק מנישר. ה-payload הוא id הקובץ שנמחק.
- שינוי הרשאה (permission\_change) - שינוי הרשאה של קובץ בשרת. ה-payload הוא id הקובץ וההרשאה החדשה ששמאים לו.
- בקשת חברות (friend\_request) - נשלחה בבקשת חברות חדשה. ה-payload הוא id הבקשה ו-id שני המשתמשים.
- חבר הוסיף (friend\_added) - אחד מן המשתמשים אישר את בקשת חברות ונוצרה חברות חדשה. ה-payload הוא id הבקשה וה-id של 2 המשתמשים.
- חבר נדחה (friend\_rejected) - אחד מן המשתמשים דחה את בקשת חברות. ה-payload הוא id הבקשה וה-id של 2 המשתמשים.

- חבר הוסר (friend\_deleted) - אחד מן המשתמשים הסיר את אחד מן החברים שלו. ה-payload הוא ה-id של שני המשתמשים.
- משתמש נוצר (user\_create) - נוצר משתמש חדש באחד מן השירותים. ה-payload הוא id המשתמש, שם המשתמש, האימייל והסיסמה.

בנוסף לכך, בכל הודעה נשלח גם הזמן הנוכחי ברגע שליחתה על מנת מעקב יותר קל אחר הודעות שנשלחות במערכת(אין באמת השפעה על ביצועי המערכת).

סוג ההודעה	נשלח מ-	נשלח אל	payload
send	השרת המרכזי	שרות אזורי	אין
receive	השרת המרכזי	שרות אזורי	אין
file_upload	שרות אזורי	השרת המרכזי	id, user_id, original_filename, stored_filename, upload_date, file_size, content
file_delete	שרות אזורי	השרת המרכזי	file_id
permission_change	שרות אזורי	השרת המרכזי	file_id, new_permission
friend_request	שרות אזורי	השרת המרכזי	request_id, from_id, to_id
friend_rejected	שרות אזורי	השרת המרכזי	request_id, from_id, to_id
friend_added	שרות אזורי	השרת המרכזי	request_id, from_id, to_id
friend_deleted	שרות אזורי	השרת המרכזי	request_id, from_id, to_id
user_create	שרות אזורי	השרת המרכזי	user_id, username, email, password

### תיאור מסכי המערכת:

מסך הרשמה - מאפשר ליצר משתמש חדש במערכת. העמוד בנוסף לכך מכיל הפניה למסך ההתחברות. במידה וההרשמה מוצלחת(פרטים אינם ריקים ותקינים), המשתמש יופנה לעמוד ההתחברות ויקבל הודעה הצלחה מתאימה.



## Create your account

Choose a username

Your email address

Create a password

Register

Already have an account? [Log In](#)

**מסך ההתחברות** – מאפשר להתחבר למערכת. עמוד זה מכיל הפניה לעמוד ההרשמה. עמוד זה הוא עמוד ברירת המחדל במכשיר ובין sessions פעיל במכשיר זה. במידה ובקשת ההתחברות תקינה(הפרטים אינם ריקים ומתאימים למשתמש קיימם), המשתמש יופנה לעמוד הבית(עמוד הצפיה בקבצים).



## Create your account

Choose a username

Your email address

Create a password

**Register**

Already have an account? [Log In](#)

**מסך הבית** - מסך זה מאפשר צפיה בקבצי המשתמש וצפיה בתונונים הקשורים לכל קובץ. מסך זה הוא מסך ברירת המחדל במידה וקיים sessions במכשיר. בנוסף, מסך זה מכיל הפניות למסך בקשות החברות, מסך החברים וכפתור ההtanתקות. במסך זה ניתן להויריד קובץ, להעלות קובץ, לשנות את הרשאות הקובץ ולמוחוק אותו מהמערכת.

SyncSphere

My Files Requests Friends List Logout

Available Files				
Filename	Upload Date	Size (bytes)	Actions	
pdf	2025-04-25 18:54	460313	<a href="#">Download</a>	<a href="#">Delete</a>
_.pdf	2025-04-30 17:41	568735	<a href="#">Download</a>	<a href="#">Delete</a>
data.txt	2025-04-30 17:48	81	<a href="#">Download</a>	<a href="#">Delete</a>
data.txt	2025-04-30 17:50	81	<a href="#">Download</a>	<a href="#">Delete</a>
sus.pdf	2025-04-30 18:56	89656	<a href="#">Download</a>	<a href="#">Delete</a>

**Effortless File Management**  
Easily upload and organize your files with SyncSphere. Simplify your workflow with quick actions.

No file chosen

**מסך בקשות החברות** - מסך זה מאפשר צפיה בבקשת החברות הקיימות ושליחת בקשה לחברות. מסך זה מכיל הפניות למסך הבית, מסך רשימת החברים וכפתור ההtanתקות. באמצעות מסך זה ניתן לאשר/לדוחות בבקשת חברות ונitin לשולח בבקשת חברות חדשה.

Dashboard Friends List Logout

**Friend Requests**

Add a friend  Send Request

Your Pending Requests  
friend @friend

Incoming Requests  
Eyalulu @Eyalulu

Friend request sent!

**מסך רשימת החברים**- מסך זה מאפשר צפייה בכל החברים של המשתמש. מסך זה מאפשר להוסיף חבר ולצפות ברשימה הקבצים שהמשתמש אישר גישה אליהם. המסך מכיל הפניה למסך הבית, מסך בקשות החברים ופתיחת התנטקיות. במידה ולחצים על כפתור צפייה ברשימה הקבצים של המשתמש, המשתמש מופנה לדף הקבצים של החבר.

Dashboard Requests Logout

**Your Friends**

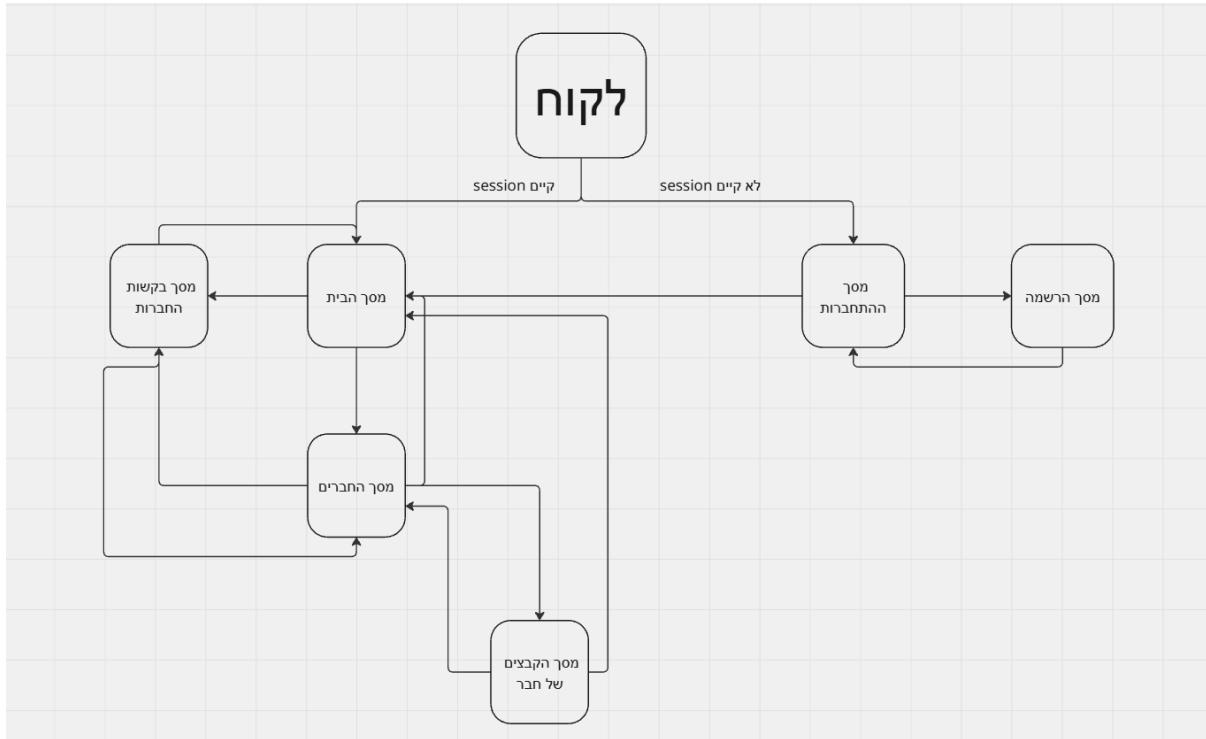
Eyalulu  
@Eyalulu  
Joined Aug 12, 2025

**מסך רשימת הקבצים של חבר**- מסך זה מאפשר צפייה בכל הקבצים המורשימים לצפייה של חבר מסוים. המשתמש יכול להויריד את הקובץ של החבר ולצפות בתונונים עלייו. המסך מכיל הפניה למסך הבית, מסך החברים ופתיחת התנטקיות.

My Files Friends Logout

**Eyalulu's Files**

Filename	Upload Date	Size	Actions
pdf	2025-04-25 18:54	460313	<input type="button" value="Download"/>



## טיאור מבני הנטוויים:

**מוד נתונים:** מוד הנתונים (sqlite3) של הפרויקט, שומר את כל הנתונים הנחוצים בשרטים האזוריים. שם מוד הנתונים הוא db.database. שטלאות הקיימות במוד הנתונים הן:  
משתמש(user) - אחראית על שמירת הנתונים הבאים הקיימים למשתמש:

- ה-**id** של המשתמש - מטיפוס int. לדוגמה, 1.
  - שם המשתמש - מטיפוס varchar(הערה להמשך, ב-sql, טיפוס זה הוא כמו string).  
לדוגמה, admin
  - אימייל - מטיפוס varchar. לדוגמה, admin@gmail.com
  - ערך ה-**hash** של הסיסמה - מטיפוס varchar. לדוגמה,  
32768:8:1\$5NPoDCFYHRpdCQpp\$0c321a4760711a9fc8325dcbf2a99e6babff  
ce55251f11e1c75389b9ce9ad3943b1fc823eedc7e4687bb1aa8e2b2dcd76ee  
(password e7442d663053af9a03a99077ced89  
תאריך יצירת המשתמש - מטיפוס datetime(תאריך). לדוגמה,  
2025-04-11T20:20:50.179491  
האחסון המשומש על ידי המשתמש(בBITS) - מטיפוס int. לדוגמה, 1000  
האחסון המקיים המתאפשר למשתמש - מטיפוס int. הערך הדיפולטיבי של שדה זה הוא  
1 גיגabytes(107374821 BITS).

בנוסף לכך, לשדות מסוימים בטבלה הוגדרו תכונות מסוימות:

- הטבלה ממוינת לפי ה-p<sub>o</sub> של המשתמשים כך שהמשתמש הראשון הוא בעל 1 p<sub>o</sub>, השני בעל 2 p<sub>o</sub> וכן הלאה. הטבלה ממוינת כך על מנת ליעיל את החיפוש כאשר מוחפשים משתמש.

- השדות username, email מוגדרים כיחודיים. כלומר, לכל משתמש חייב להיות שם משתמש ואי מייל ייחודי.

קובץ(file) - אחראית על שימרת הנתונים הקשורים לקבצים המאוחסנים במערכת:

- ה-`id` של הקובץ, - מטיפוס int
- ה-`id` של המשתמש אליו שייך הקובץ - מטיפוס id. לדוגמה, 1
- שם הקובץ המקורי - מטיפוס varchar. לדוגמה, test.txt
- שם הקובץ באחסון - מטיפוס varchar (על הסיבה לשני שמות שונים לכל קובץ - המקורי ובאחסון יורחב בהמשך תיאור המערכת). לדוגמה,  
08eef1e5d62345319b3600912cac35ff.pdf
- תאריך העלאה - מטיפוס datetime. לדוגמה, 18:54 2025-04-11
- גודל הקובץ(בבייטים) - מטיפוס int. לדוגמה, 1000
- הרשות הצפיה בקובץ(private, friends\_only, public) - מטיפוס varchar. לדוגמה, private

בנוסף לכך, לשדות מסוימים בטבלה הוגדרו תכונות מסוימות:

- הטבלה ממיינת לפי ה-`id` של הקבצים כך שהקובץ הראשון הוא בעל 1 id, השני בעל 2 id וכן הלאה. הטבלה ממיינת כך על מנת ליעל את החיפוש כאשר עוברים על הקבצים.
- העמודה `id_user` בנוסף לכך מצביעה על עמודת ה-`id` בטבלת המשתמש(`user`). בנוסף לכך, כאשר משתמש נמחק מהטבלה `user`, כל הקבצים בעלי ה-`id user` יימחקו גם הם. הגדרנו זאת על מנת לקשר בקלות בין כל קובץ לבניו.

בקשת חברות(friend\_request) - אחראית על שימרת חברות במערכת ונתונים הקשורים אליהן:

- ה-`id` של הבקשה - מטיפוס int. לדוגמה, 1.
- ה-`id` של שולח הבקשה - מטיפוס int. לדוגמה, 1.
- ה-`id` של יעד הבקשה - מטיפוס int. לדוגמה, 1.
- סטטוס הבקשה('ממתינה', 'מאושרת', 'נדחתה') - מטיפוס varchar. לדוגמה, pending.
- תאריך יצירת הבקשה - מטיפוס datetime. לדוגמה, 18:54 2025-04-11.

בנוסף לכך, לשדות מסוימים בטבלה הוגדרו תכונות מסוימות:

- העמודות של שולח הבקשה, נמען הבקשה מצבעות על עמודת ה-`id` בטבלת המשתמש(`user`). בנוסף לכך, כאשר משתמש נמחק מהטבלה `user`, כל חברות הקשורות אליו יימחקו גם הן. הגדרנו זאת על מנת לקשר בקלות בין כל בקשה למשתמשים הקשורים אליה.

חברות(friendship) - אחראית על שימרת חברות במערכת:

- ה-`id` של חברות - מטיפוס int. לדוגמה, 1.
- ה-`id` של המשתמש הראשון - מטיפוס int. לדוגמה, 1.

- ה-p<sub>o</sub> של המשתמש השני - מטיפוס int. לדוגמה, 1.
- תאריך אישור הבקשה - מטיפוס datetime. לדוגמה, 18:54 2025-04-11.

בנוסף לכך, לשדות מסוימים בטבלה הוגדרו תכונות מסוימות:

- הטבלה ממיונית לפי ה-p<sub>o</sub> של בקשות החברות כך שבקשת החברות הראשונה הוא בעל p<sub>o</sub> 1, השנייה בעלי 2 ון הלאה. הטבלה ממיונית כך על מנת ליעיל את החיפוש כאשר מוחפשים עוברים על הבקשות.
- העמודות של שני המשתמשים החברים מצביעות על עמודת ה-p<sub>o</sub> בטבלת המשתמשים(user). בנוסף לכך, כאשר משתמש נמחק מהטבלה user, כל החברות הקשורות אליו ימחקו גם הן. הגדרנו זאת על מנת לקשר בקלה בין כל חברות למשתמשים הקשורים אליה.

## סבירות חולשות ואיומים:

המתפקידות/חולשות המרכזיות שלדעתינו שווות התיאחות הן:

1. **sql injection** - במידה ולא מתגוננים מכך בהתאם, מבנה הנתונים עלול להיות רגיש ל-sql injection. עקב כך שאני משתמש ב-sql alchemy לעובדה עם מסד הנתונים בקוד, המערכת מוגנת מ-sql injection עקב כך ש-sql alchemy משמש בשאלות sql מובנות כך שהסיכון להזרקת שאלתה זדונית פוחת

2. **-cross site request forgery** - התגוננות מתקפה זו באמצעות טוקן ייחודי. בכל בקשה לשרת, הטוקן הייחודי צריך להישלח לשרת. למעשה, במידה ואטר זמני מנסה לבצע פעולה במערכת, ללא הטוקן הייחודי (שאין לו גישה אליו) לא ניתן לבצע פעולות בשרת.

3. **- session hijacking** - על מנת למנוע מתקפה זו, הגדרתי את ה-e-session cookie כך שלא ניתן לגשת אליה מצד שלישי, לא ניתן לגשת אליה באמצעות קוד javascript(<http://> קוד only) והן נשלחות רק על גבי <https://>

4. **MITM** - בrama עקרונית, כיוון שהמערכת משתמשת ב-ssl, המערכת אמורה להיות מוגנת מ-mitm. עם זאת, עקב כך שתעודת ה-ssl לא מוכרת רשמית ע"י אף רשות(על מנת שהיא מוכרת זהו תהליך שצורך זמן וכסף) ובמהלך לחיצת היד ב-ssl לא מתבצע אימוץ שהשרת הוא אכן השירות האמיתי כך שניתן עדין להתחזות לשרת. עם זאת, במידה ובאמת הייתה משתמש בתעודת מוכרת, הפרויקט היה מוגן מתקפה זו.

5. **bruteforce**- במהלך תהליך ההתחברות, ישנה הגבלה של 5 ניסיונות התחברות שגויים. לאחר 5 ניסיונות, לא ניתן להתחבר למערכת מאותו מכשיר במשך 15 דקות. עקב מנגנון זה, מתאפשר bruteforce אינה אפשרית.

6. **DDoS**- בנגד לאיומים שהוצעו מוקדם, נגד איום זה לא מימושי הגנה משמעותית כרשות מקצועות מסווג זה הן בהחולט חולשה קיימת. בנוסף לכך, ניתן לבצע מספר פעולות רב במערכת(אין הגבלה של כמות פעולות לפרק זמן מסוים) כך שניתן לנצל את כל משאבי המערכת ולמנוע תפקוד תקין.

**האגנה במערכת**- על מנת להציג את התקשרות במערכת, השתמשתי ב-AES. במהלך העברת המידע עצמו, משתמשים בהצפנה סימטרית(Sצפיפות-B-AES). על מנת לפתור את בעיית שיתוף המפתחות, המערכת משתמשת בפרוטוקול דיפי הלמן.

## מימוש הפרויקט:

### סידרת המודולים, המחלקות וקשרי הgomelin ביניהם:

#### סידרת המודולים המובאים:

- so - עבודה עם מערכת הקבצים(חיבור מיקום של קבצים, מציאת תיקייה העבודה הנוכחית וכו').
- threading - עבודה עם מספר threads במקביל
- werkzeug(flask) - יכולת המודולים תחתוי כגון flask-wtf等工作 (וכו') - מסגרת עבודה לניהול נתונים, html rendering, עיבוד טפסים, ניהול sessions וגיבוב סיסמאות.
- alchemy - עבודה עם מסד הנתונים datetime - טיפול בתאריכים וזמינים
- socket - יצרת חיבור בין 2 רכיבים במערכת(Sצפיפות השרת המרכזי ושרת אזורי)
- ssl - יצרת חיבור מאובטח בין 2 סוקטים
- json - שליחת מבני נתונים בין סוקטים(סריאליזציה)
- time - הייתה התכוון במרקמים מסוימים(למנוע מלולאה לרוץ תמיד)
- pwind - יצרת מזהים ייחודיים לשמרות הקבצים במערכת

## מודולים שפיתחת:

### User .1

#### תפקיד המחלקה- מייצגת משתמש קיימם במערכת.

##### תכונות המחלקה:

- ❖ pi - מזהה ייחודי של המשתמש.
- ❖ username - שם המשתמש של המשתמש, גם הוא ייחודי.

- ❖ email - האימייל של המשתמש. גם הוא ייחודי לכל משתמש.
- ❖ password\_hash - ערך ה-hash של סיסמת המשתמש (המערכת שומרת את ערך ה-hash של הסיסמה במקומם הסיסמה עצמה מטעני אבטחה).
- ❖ created\_at - זמן יצירת המשתמש (תאריך ושעה).
- ❖ used\_storage - כמות האחסון שהמשתמש מנצל במערכת ( נשמר בבייטים ).
- ❖ storage\_quota - מכוסת האחסון שהמשתמש יכול לנצל.

#### **פעולות המחלקה:**

- ❖ (password\_set(self, password) - פעולה המקבלת סיסמה, מבצעת עליה את פונקציית hash והשומרת את הערך בתוכונה password\_hash).
- טענת כניסה - password, משתנה מסוג string שמייצג את הסיסמה של המשתמש במחלקה.
- טענת יציאה - הפעולה מבצעת על הסיסמה את פונקציית hash ושומרת את הערך בתוכונה password\_hash. הפעולה לא מחזירה כלום.
- ❖ (check\_password(self, password) - פעולה מקבלת סיסמה ובודקת האם הסיסמה זהה לסיסמה של המשתמש).
- טענת כניסה - password, משתנה מסוג string שמייצג סיסמה שרוצים לבדוק האם היא של המשתמש.
- טענת יציאה - הפעולה מבצעת את פונקציית hash על הסיסמה, משווה אותה לערך hash של סיסמת המשתמש ומחזירה אמת במידה והסיסמאות זהות ושקר אם אחרת.

## **File .2**

### **תפקיד המחלקה-** מייצגת נתונים של קובץ מסוים (לא כולל תוכן הקובץ עצמו).

#### **תכונות המחלקה:**

- ❖ id - מזהה ייחודי של הקובץ.
- ❖ user\_id - ה-id של בעל הקובץ.
- ❖ stored\_filename - השם הייחודי של הקובץ במערכת הקבצים.
- ❖ original\_filename - השם המקורי של הקובץ (השם שבו המשתמש העלה את הקובץ).
- ❖ upload\_date - שעת ותאריך העלאת הקובץ למערכת.
- ❖ file\_size - גודל הקובץ בבייטים.
- ❖ permissions - הרשאות הגישה לקובץ(פרט, חברים בלבד, ציבור).

#### **פעולות המחלקה:**

אין למחלקה פעולות מיוחדות.

## FriendRequest .3

**פקיך המחלקה**- מייצגת בקשת חברות בין 2 משתמשים במערכת.

### תכונות המחלקה:

- ❖ id - מזהה ייחודי של הבקשה.
- ❖ from\_user\_id - ה-id של שולח הבקשה.
- ❖ to\_user\_id - ה-id של מקבל הבקשה.
- ❖ status - סטטוס הבקשה(ממתין לאישור, אוישר, נדחה)
- ❖ created\_at - שעת ותאריך ייצרת שליחת בקשת החברות

### פעולות המחלקה:

אין למחלקה פעולה מיוחדות.

## Friendship .4

**פקיך המחלקה**- מייצגת חברות בין 2 משתמשים במערכת.

### תכונות המחלקה:

- ❖ id - מזהה ייחודי של החברות
- ❖ user\_id - ה-id של המשתמש הראשון.
- ❖ friend\_id - ה-id של המשתמש השני.
- ❖ created\_at - שעת ותאריך ייצרת החברות.

### פעולות המחלקה:

אין למחלקה פעולה מיוחדות.

## FileManager .5

**פקיך המחלקה**- מכבצת את ניהול הקבצים (שמירת קבצים, מחיקת קבצים, הרשות גישה וכו') למחלקה.

### תכונות המחלקה:

- ❖ upload\_folder - נתיב התקינה בה אחסנים את הקבצים שמשתמשים מעליים למערכת.

### פעולות המחלקה:

- ❖ allowed\_file(self, filename) - פעולה המקבלת קובץ ובודקת אם סוג קובץ זה מותר.
- טענת כניסה - filename, משתנה מסוג string המציג שם של קובץ (כולל סימנות הקובץ).
- טענת יציאה - הפעולה בודקת האם סימנת הקובץ מורשית במערכת (ישן סימנות קבצים שלא מותרות במערכת מטעמי אבטחה) ובמידה והקובץ מורשע מוחזירה אמת, אחרת שקר.

❖ `generate_unique_filename(self, filename)` - פעולה המקבלת שם של קובץ ומיצרת לו שם ייחודי.

- טענת כניסה - `filename`, משתנה מסוג `string` המציג שם של קובץ (כולל סימן הקובץ).

• טענת יציאה - הפעולה מייצרת לקובץ שם ייחודי (`id`), מוסיפה לשם הייחודי את הסימן ומחזירה את השם הייחודי.

❖ `save(self, file_storage, user)` - פעולה המקבלת קובץ ומשתמש, שומרת את הקובץ

- טענת כניסה - `file_storage`, אובייקט של הקובץ שהועלה דרך `flask.request.user`, אובייקט מסוג `user` המציג את המשתמש שהעלה את הקובץ.

• טענת יציאה - הפעולה שומרת את הקובץ בתיקייה, מעדכנת את מסד הנתונים (מוסיפה `file` חדש), מעדכנת את השדה `used_storage` אצל המשתמש, מכניסה את האירוע לטור `queue_changes` ומחזירה אובייקט מסוג `file` של הקובץ

❖ `list_user_files(self, user)` - פעולה המקבלת משתמש ומחזירה את כל הקבצים השייכים למשתמש.

- טענת כניסה - `user`, תם.

❖ `get_file_record(self, file_id)` - פעולה המקבלת `id` של קובץ ומhzירה את נתוני הקובץ מסד הנתונים.

- טענת כניסה - `file_id` משתנה מסוג `int` המציג `id` של קובץ.

• טענת יציאה - הפעולה מבצעת שאילתת `sql` על מנת לקבל את הקובץ שה-`id` שלו זהה ל-`id` המתבקש ומחזירה אובייקט מסוג `user` המציג משתמש.

• טענת יציאה - הפעולה מבצעת שאילתת `sql` על מנת לסנן את הקבצים השייכים למשתמש ומחזירה או את נתוני הקובץ מסד הנתונים במידה והוא קיים, ו-`None` אם לא.

❖ `delete_file(self, file_record, user)` - פעולה המוחקת קובץ מהמערכת וمعدכנת את מסד הנתונים.

- טענת כניסה - `file_record`, נתונים הקובץ מסד הנתונים ו-`user`, אובייקט מסוג משתמש שמייצג משתמש שרוצה למחוק קובץ.

• טענת יציאה - הפעולה מוחקת את הקובץ הקיימים המערכת הקבצים (במידה ולמשתמש יש גישה למחוק את הקובץ), מסירה את רשותת הקובץ מסד הנתונים, מכניסה את האירוע לטור `queue_changes` ומחזירה `true` במידה והמחיקה הצליחה.

❖ `update_permissions(self, file_record, new_permissions, user)` - פעולה המקבלת קובץ, משתמש הרוצה לשנות הרשאה של קובץ והרשאה חדשה וمعدכנת את הרשותת הקובץ.

- טענת כניסה - `file_record`, נתונים הקובץ מסד הנתונים, `user`, אובייקט מסוג משתמש שמייצג משתמש לשנות את הרשותת הקובץ ו-`new_permissions`, משתנה מסוג `string` שמייצג את הרשותה החדשה לקובץ.

- טענת יציאה - הפעולה משנה את הרשות הקובץ (במידה ומשתמש יש גישה לעשות זאת), מעדכנת את נתוני הקובץ בסיסד הנתונים, מכניסה את האירוע לתוך changes\_queue ומחזירה את רשומת הקובץ המעודכנת.
- ❖ `user.access_allowed(self, file_record)` - פעולה הבודקת אם למשתמש יש גישה לקובץ.
- טענת כניסה - `file_record`, נתונים הקובץ מסיסד הנתונים ו-`user`, אובייקט מסווג משתמש שמייצג משתמש.
- טענת יציאה - הפעולה בודקת אם למשתמש יש גישה לקובץ, מחזירה אמת במידה וכן ושער אם אחרת.

## FriendManager .6

**תפקיד המחלקה** - מerbצאת את ניהול החברויות, בקשות החברות ופעולות הקשורות לכך (יצירת חברות, מחיקת חברות וכו').

### תכונות המחלקה:

אין למחלקה זו תכונות.

### פעולות המחלקה:

- ❖ `user.send_request(self, from_user, to_user)` - פעולה מקבלת שני משתמשים ושולחת בקשה חברות אחד לשני.
  - טענת כניסה - `from_user`, אובייקט מסווג `user` המייצג את שולח הבקשה ו-`to_user`, אובייקט מסווג `user` המייצג את מקבל הבקשה.
  - טענת יציאה - הפעולה בודקת שבבקשת החברות חוקית (לא קיימת כבר בקשה חברות בינם, המשתמשים לא חברים, שני המשתמשים הם שונים), מעדכנת את מסד הנתונים, מכניסה את האירוע לתוך changes\_queue ומחזירה מופע של friendrequest
- ❖ `user.get_incoming_requests(self)` - פעולה מקבלת משתמש ומחזירה רשימה של בקשות החברות שנשלחו אל המשתמש.
  - טענת כניסה - `user`, אובייקט מסווג `user` המייצג את המשתמש שורצים לראות את בקשות החברות שנשלחו אליו.
  - טענת יציאה - הפעולה מבצעת שאילתת `qfs` על מנת לקבל את כל בקשות החברות שהמשתמש שאליהן נשלח הבקשה הוא `user` ושהסטטוס שליהן הוא "מתינונת לאישור", ומחזירה את הבקשות הללו.
- ❖ `user.get_outgoing_requests(self)` - פעולה מקבלת משתמש ומחזירה רשימה של בקשות החברות שהמשתמש שלח.
  - טענת כניסה - `user`, אובייקט מסווג `user` המייצג את המשתמש שורצים לראות את בקשות החברות שהוא שלח.
  - טענת יציאה - הפעולה מבצעת שאילתת `qfs` על מנת לקבל את כל בקשות החברות שהמשתמש שלח את הבקשה הוא `user` ושהסטטוס שליהן הוא "מתינונת לאישור", ומחזירה את הבקשות הללו.

❖ - פועלה מקבלת בקשת חברות respond\_request(self, request\_id, accept)

והתשובה אליה ומשנה את הסטטוס של הבקשה.

- טענת כניסה - id\_=request\_id, משתנה מסווג זה מייצג את ה-id של הבקשה שורצים לשנות את הסטטוס שלה ו-accept, המשתנה בוליאני המחזיק true במידה והבקשה אושרה ו-false במידה והיא נדחתה.

טענת יציאה - הפעולה מעדכנת את סטטוס הבקשה בסיסד הנתונים, במידה והבקשה אושרה מעדכנים את טבלת friendship, מכניסה את האירוע לטור friendrequest queue\_change\_true ומחזירה מופיע של changes\_queue.

❖ - פועלה מקבלת משתמש ומחזירה את רשימת החברים שלו get\_friends(self, user)

- טענת כניסה - user, אובייקט מסווג user המייצג משתמש שורצה את רשימת חברים.

טענת יציאה - הפעולה מבצעת שאילתת sqs על מנת לקבל את כל המשתמשים חברים של המשתמש ומחזירה רשימה של כל החברים.

❖ - הפעולה מקבלת שני משתמשים ומסירה את remove\_friend(self, user, to\_user)

החברות ביניהם.

- טענת כניסה - user, אובייקט מסווג user המייצג את המשתמש שמסיר את הבקשה ו-to\_user, אובייקט מסווג user המייצג את המשתמש שמוציא מהחברים.

טענת יציאה - הפעולה מוחקת מסיס הנתונים את החברות בין שני המשתמשים, מכניסה את האירוע לטור queue\_change\_true במידה והחברות הוסרה בהצלחה.

## **בעיות אלגוריתמיות - פירוט**

1. אתחל דוקא מהבעיה השנייה (בעיית התלות בשרת אחד בלבד). הפתרון שלי לבעה זו היה יצירת שרת מרכזי אחד והקמת מספר שירותים אזוריים שאיתם הלקוח יוצר אינטראקציה. החיבור לשרת זה מתבצע כאשר מרים את app.config, כאשר בקובץ זה התכנית מרים גם את התוכן של app.py וגם את החיבור לשרת המרכזי (באמצעות multithreading).

```

3     # Launch the background sync thread exactly once
4     print("[Sync] Launching background sync thread...", flush=True)
5     threading.Thread(target=connect_to_server, daemon=True).start()
6
7     # Start the Flask HTTPS server (self-signed cert for dev)
8     app.run(
9         host='0.0.0.0',
10        debug=False,                      # disable debugger in production
11        ssl_context=(certfile, keyfile)
12    )

```

בקובץ config (יוסבר עליו בהמשך במדריך למשתמש) מוגדר תור (queue) המכיל את השינויים השונים המתבצעים בשרת איזורי.

changes\_queue = Queue()

לטור זה מכנים אירוע שמתעדכן במאסן הנתונים של השירות. האירועים הללו הם: העלאת קובץ, מחיקת קובץ, שינוי הרשאה של קובץ, יצירת משתמש, רשימת בקשות חברות, תגובה לבקשת חברות, הסרת חבר. במימוש של כל אחת מן הפעולות הללו, מכנים את האירוע הרלוונטי לתור. לדוגמה, בפעולה להסרת הקובץ ממאסן הנתונים:

```
def delete_file(self, file_record, user):
    """ Delete a file from disk and remove its DB record. Only the file owner may delete. Enqueues a file_delete sync event. """
    if file_record.user_id != user.id:
        # Prevent unauthorized deletions
        raise PermissionError("You are not authorized to delete this file.")
    # Remove file from disk
    file_path = os.path.join(self.upload_folder, file_record.stored_filename)
    if os.path.exists(file_path):
        os.remove(file_path)
    # Remove DB record
    db.session.delete(file_record)
    db.session.commit()

    # Notify other servers of deletion
    changes_queue.put({
        "type": "file_delete",
        "file_id": file_record.id,
        "timestamp": datetime.now().isoformat()
    })
    return True
```

ניתן לראות בתחתית הפונקציה, מכנים לטור התוור את הנתונים הרלוונטיים לשינוי זה. פעולה זו מתבצעת בכל אירוע הדורש שמירה במאסן הנתונים במערכת. דוגמה נוספת לכך ניתן לראות בשינוי הרשאות של קובץ:

```
def update_permissions(self, file_record, new_permissions, user):
    """ Change the permission of a file (private or public) Only the owner may change permissions. Enqueues a permission_change sync event. """
    if file_record.user_id != user.id:
        # Prevent unauthorized permission changes
        raise PermissionError("You are not authorized to change permissions for this file.")
    file_record.permissions = new_permissions
    db.session.commit()

    # Notify other servers of permission change
    changes_queue.put({
        "type": "permission_change",
        "file_id": file_record.id,
        "new_permissions": file_record.permissions,
        "timestamp": datetime.now().isoformat()
    })
    return file_record
```

ובהוסף קובץ למערכת:

```

# Read file content for sync event
with open(file_path, 'rb') as f:
    content = f.read()

# Enqueue a file_upload event for synchronization
changes_queue.put({
    "type":      "file_upload",
    "payload": {
        "id":                  file_record.id,
        "user_id":             file_record.user_id,
        "stored_filename":     file_record.stored_filename,
        "original_filename":   file_record.original_filename,
        "upload_date":         file_record.upload_date.isoformat(),
        "file_size":           file_record.file_size,
        "permissions":         file_record.permissions,
        "content":             content
    },
    "timestamp": datetime.now().isoformat()
})

```

בנוסף לשרת-ה-HTTP שרצה ע"י הריצת הקובץ `app.py`, רץ הקובץ `sync.py` שהוראי על תקשורת עם השרת המרכז. בקובץ זה, השרת האזרוי ממחכה להודעה המורה עליו לשלוח לשרת המרכז את השינויים שנעשו בשרת זה או הודעה המודיעה על השינויים שנערכו בשרתים אזרויים אחרים.

```

message_type = received_message.get('type')
if message_type == 'send':
    send_changes(sock)
elif message_type == 'receive':
    receive_changes(received_message)

```

אתually מהאפשרות הראשונה, המורה על שליחת השינויים לשרת. באפשרות זו, השרת האזרוי עובר על כל אירוע שקרה מאז הסyncronization האחרון, מצרך אותו ל-`list` ושולח לשרת המרכז. עם זאת, במידה והאירוע שקרה הוא קובץ חדש שהועלה, השרת צריך לבצע עוד מספר פעולות. כאשר שומרים את האירוע של הוספה קובץ בתור, שומרים את תוכן הקובץ כב"יטים. על מנת לשלוח את המידע בונגוע לכל אירוע, בחרתי להשתמש בספרייה `chardis`, וספרייה זו עבדת עם מחרוזות. לפיכך, צריך להמיר את תוכן הקובץ למחרוזת של ביטים. בשביל כך, ממיר את תוכן הקובץ למחרוזת, מעדכנים את המידע על אירוע ושולחים את המידע לשרת המרכז. בסך הכל, פעולה זו נראית כך:

```
# If it's a new file upload, Base64-encode raw bytes for JSON transport
if change['type'] == 'file_upload':
    p = change['payload'].copy()
    raw = p.pop('content') # remove binary data from payload
    # encode bytes to UTF-8 string so it can be embedded in JSON
    p['content'] = base64.b64encode(raw).decode('utf-8')
    events.append({
        'type': change['type'],
        'payload': p,
        'timestamp': change['timestamp']
    })
else:
    # other event types can be sent as-is
    events.append(change)
```

עכשו, נעבור לאפשרות השנייה, שהיא קבלת השינויים שחלו בשרתים אחרים. באפשרות זו השרת האזורי מקבל את השינויים מהשרת המרכזי (על תהליך זה ארכיב בהמשך), ובהתאם לכל שינוי מבצע את הפעולות הנדרשות. לדוגמה, במידה וنمחק קובץ, השרת האזורי מוחק את הקובץ מסודן הנטוניים.

```
elif event_type == 'file_delete':
    rec = file_manager.get_file_record(ev['file_id'])
    if not rec:
        print(f"[Sync] Warn: no file record for deletion ID {ev['file_id']}", flush=True)
        continue
    user = User.query.get(rec.user_id)
    file_manager.delete_file(rec, user)
```

במקרה ושינוי מסוים הוא הוספה של קובץ, צריך לעשות את התהליך הפוך מהתהליך שביצעונו בשליחת השינויים לשרת המרכזי (צריך להמיר את תוכן הקובץ ממחוזת לביטים) והוא ליצור את הקובץ גם במערכת הקבצים. תהליך זה נראה כך:

```
if event_type == 'file_upload':
    # Decode and write file bytes
    payload = ev['payload']
    data = base64.b64decode(payload['content'])
    dest = os.path.join(UPLOAD_FOLDER, payload['stored_filename'])
    os.makedirs(os.path.dirname(dest), exist_ok=True)
    with open(dest, 'wb') as f:
        f.write(data)
```

עכשו, נעבור לדzon בפתרון הבעיה מצד השרת המרכזי. מטרתו של השרת המרכזי הוא לסנכרן את השינויים המתרחשים בכל שרת אזורי כך שבמידה ומתבצעת פעולה בשרת מרכזי אחד, יהיה אפשרי לגשת לשרת אזורי אחר ואוותה פעולה שהתבצעה שם תתבצע גם בשרת זהה. השרת המרכזי מażין לשרתים אזוריים וכאשר אחד כזה מתחבר, נוצר thread חדש לשרת המרכזי המטפל בשרת אזורי זה.

## SyncSphere

```
while True:
    try:
        raw_conn, addr = server_sock.accept()
        # Perform TLS handshake on the new connection
        try:
            conn = context.wrap_socket(raw_conn, server_side=True) #Wraps the socket w
            print(f"[GrandServer] Regional connected (TLS): {addr}", flush=True)

        except ssl.SSLError as e:
            print(f"[GrandServer] SSL handshake failed with {addr}: {e}", flush=True)
            raw_conn.close()
            continue

        with clients_lock:
            clients.append(conn)
        # Replay missed events for reconnects
        send_history(conn)
        threading.Thread(target=client_handler, args=(conn,), daemon=True).start() #st
```

כאשר מגיע הזמן לسانכרן את העדכנים, השרת המרכזי שולח הודעה לכל השירותים המתחברים אליו המצביעה על כך. פעולה זו רצתה בתוך thread נפרד שתפקידו הוא לדאוג לכך.

```
y.
time.sleep(SYNC_INTERVAL) #Sleeps until the next sync process.
print("[GrandServer] Requesting changes from all regionals...", flush=True)
packet = json.dumps({'type': 'send'}) + '\n'
data = packet.encode('utf-8')

with clients_lock:
    for conn in list(clients):
        try:
            conn.sendall(data)
```

בintéיימ, שאר ה-threads מאזינים להודעות מהשירותים האזוריים הכוללות את השינויים.

```

"""Read incoming 'changes' messages
addr = conn.getpeername() #Gets the
f = conn.makefile('r') #Wraps the s
try:
    for line in f:
        #Continiously reads each in
        line = line.strip() #Ignoring
        if not line:
            continue

```

התפקיד של השורה השנייה פה הוא לעטוף את הsocket כסוג של קובץ. היתרונות של עטיפה זו היא שימוש במהלך קריאת נתונים במקומ להשתמש בrecv וניתן לעבור באיתרציה על כל שורה בקובץ. הסיבה לכך רלוונטי לנו היא שכאשר שרת אזורי שולח את השינויים, הוא מפריד כל שינוי ח'ו שמייצג שורה חדשה. בשימוש בדרך זו ניתן בקלות להפריד בין כל שינוי ובצע פעולות הקשורות אליו. כאשר השירות מקבל את כל השינויים משרת אזורי, הוא שולח הודעה הדומה להודעת broadcast לשאר השירותים המוחברים לשרת המרכזי.

```

packet = json.dumps({'type': 'receive', 'events': events}) + '\n'
data = packet.encode('utf-8')

with clients_lock:
    for conn in list(clients):
        if conn is exclude:
            continue
        try:
            conn.sendall(data)

```

פתרון זה מקטין את התלות בשרת אזורי כלשהו. במידה ושרת אזורי מסוים קורם/מתנתק, עדין אפשר להתחבר לשרת אזורי אחר ולהשתמש במערכת. על מנת שירותים אזוריים יוכל לזרע גם לאחר שהם קורסו (בלי שייהיו חסרים בהם שינויים מסוימים), כאשר שירות אזורי מתחבר לשרת המרכזי, השירות המרכזי דואג לעדכן את השירות האזורי בכך. השירות המרכזי דואג לשומר כל שינוי שהתרחש ב-5 הסנכרונים האחרונים (מספר זה הוא שרירותי, ניתן לשנות אותו במידת הצורך). כאשר מתחבר שירות אזורי למערכת, השירות המרכזי דואג לשולח לו את כל השינויים שקרו בזמן זה.

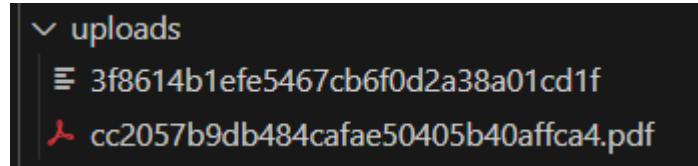
```
def send_history(conn):
    """Sends the last 5 events to a regional_server."""
    for events in history:
        packet = json.dumps({'type': 'receive', 'events': events}) + '\n'
        try:
            conn.sendall(packet.encode('utf-8'))
```

הבעיה המרכזית שנותרה לפתור זה היא מקרה שבו השרת המרכזי קורט. במקרה זה, לא ניתן למסנן את השינויים בין השירותים האזרחיים אך כל שרת אזרחי פועל כרגע חוץ ממנו שנוגע למסנן השירותים האחרים. על מנת לפתור בעיה זו, ניתן ליצור מערכת של כמה שירותים מרכזיים שמתקשרים ביניהם, כך שבמקרה ושרת מרכזי אחד נופל, השירותים המרכזיים האחרים יכולים לכפות עליהם. את הפתרון זהה לא מימושי, אך פתרון זה הוא יחסית פשוט ודורש הקמה של רשת Soketים קטן (ולא עוד לוגיקה/אלגוריתם מיוחדת). הfonקציות המלאות: השירות האזרחי:

2. עבור לבעה שנשארה, שהיא מבנה מערכת הקבצים. בכל שרת אזרחי ישנה תיקייה בשם `sends` שכוללת את הקבצים המקוריים על השירות (הקבצים של כל המשתמשים מופיעים בתיקייה זו). על מנת שלא יהיו התנגשויות בין שמות קבצים (ייתכן מצב של שני משתמשים שונים יש קובץ בעל אותו שם), לכל קובץ יש שם מיוחד. את השם המיוחד הזה מייצרים על ידי פונקציית `uuid` (מצאה ייחודי לכל קובץ שאורך 128 ביטים). הסיכוי להתנגשות בין שמות של שני קבצים שווים ל-0 (אחד לשתיים בחזקת 128). בנוסף לכך, לכל קובץ מצרפים למצאה זה את סימנת הקובץ.

```
def generate_unique_filename(self, filename):
    """ Generate a unique filename using UUID4 and preserve the file extension """
    ext = filename.rsplit('.', 1)[1] if '.' in filename else ''
    # Combine a random hex string with the original extension
    unique_name = f"{uuid.uuid4().hex}.{ext}" if ext else uuid.uuid4().hex
    return unique_name
```

כתוצאה לכך, כך תיקיות הקבצים עשוייה להיראות:



כאשר רוצים לגשት לקובץ מסוים במערכת, במסד הנתונים רשום לכל קובץ גם את השם המקורי שלו וגם את השם שבו הוא נשמר בזיכרון, וכך ניתן למצוא את השם המקורי של הקובץ באמצעות השם במערכת הקבצים ולהפוך.

## **תיאור הבדיקות במערכת:**

מספר הבדיקה	שם הבדיקה	מה הבדיקה אמרה לבודק	איך מתוכנים לבודק	תוצאות הבדיקה
1	התחברות והרשמה למערכת	האם ניתן להתחבר ולהירשם למערכת	הרשמה למערכת ולאחר מכן ניסיון התחברות	עובד
2	טיפול בלקוח אחד	האם המערכת מסוגלת לטפל בפעולות הבסיסיות של לקוח אחד	ביצוע הפעולות הבאות - העלאת קובץ, מחיקת קובץ, שינוי הרשות קובץ, הורדת קובץ, שילוח בקשות חברות, אישור בקשה וצפיה בקבצים של חבר	תחילת רוב הפעולות עברו בהצלחה חוץ מהפעולה של צפיה בקבצים של חבר. עקב כר נאלצתי לשנות את הבדיקה לאיזה קבצים משתמש יכול לגשת ולאחר זאת זהעובד
3	התחברות ממזכיר חיצוני	האם המערכת מסוגלת לטפל בפעולות הבסיסיות של לקוח אחד	ביצוע אותן פעולות על גבי מחשב שונה כאשר השרת רץ על אותו מזכיר	עובד

עובד	לאחר ביצוע הפעולות, אsegור את השרת ואבדוק אם הקבצים הרלוונטיים + הפעולות נשמרו	האם השינויים שנעשו נשמרים במסד הנתונים ומערכת הקבצים	טיפול במסד הנתונים ומערכת הקבצים	4
תחליה לא עבד עקב כר שהתעודה לא מיהינה והמחשב שלו חסם את האתר. על מנת לפטור את זה, היה צריך לאשר את התעודה במכשיר באמצעות שימוש certmgr.msc, אישרתי את התעודה אז המערכת חזרה לעבוד. לאחר מכן, ניתן הסנהפה, ניתן היה לראות שהמערכת מוצפנת	הסנהפה באמצעות wireshark של תעבורת המערכת	האם המערכת מסוגלת להציג את התקשרות	הצפנה המ מערכת	5

עובד	אנסה להיכנס למערכת מספר פעמים ברציפות ואראה אם ישנה חסימה לאחר מספר נסיגות מסוימות	האם המערכת מסוגלת להתמודד עם <b>bruteforce</b>	ניסיונות התחברות מתמשכים	6
תחילת הסyncron לא עבד עקב כרiscal שרת אזרוי התחבר פעמיים לשרת המרכז. הסיבה לכך היא reload flask מרייז 2 ,threads וכתוצאה מכך נוצרים threads 2 שמתחברים לשרת המרכז. כאשר מבטלים את ה-reloader נוצר רק thread אחד והמערכת עובדת	ארץ שני שרתים אזרוים משני מכשירים שונים, אבצע שינוי באחד מהם ואראה אם הוא מתעדכן בשני	האם מערכת מצלהן לשונרן שינוי בין מספר שרתים אזרויים הפועלים בו בזמןית	סנכרון שינויים בין שרתים אזרויים	7

עובד	באמצעות כלי בשם <b>sqlmap</b> אריז שאלות אql זדוניות ואראה אם המערכת מוגנת מכך	האם המערכת <b>מוגנת מ- sql injection</b>	<b>sql injection</b>	8
עובד	אריז את השרותים האזוריים ואכבה את שרות השרות המרכזיזי, ולאחר מכן אבל ערך פעולות על מנת לראות אם השירות האזוריז מתפקיד	האם השרותים האזוריים מסוגלים לפעול במצב שבו שירות המרכזיזי קורס	קריישה של השרות המרכזיזי	9
עובד	ביצוע מספר שינויים בשרות אזוריז ולאחר זמן מה הפעלה של שירות אזוריז חדש. ואראה אם השינויים שהתרחשו בשירות האזוריז הוותמעו גם בשירות החדש	האם השירות אזוריז שהתחבר באחור/אחרי קריישה מסוגל להשלים את השינויים שהתרחשו בזמן זה	התחברות של מאחורית של שירות אזוריז	10

**בדיקות נוספות שביצעתו למערכת:**

מספר הבדיקה	שם הבדיקה	מה הבדיקה אמרה לבדוק	איך מתקנים לבדוק	תוצאות הבדיקה
11	גישה לקובץ ציבורי	בדיקה האם משתמש שAINO הביעים של קובץ יכול להוריד אותו אך לא יכול לשנות כלום לגביו	העלאת קובץ, הגדרתו כציבורי והתחברות משתמש אחר שהוא חבר של המשתמש	עובד
12	שימוש במכשיר טלפון	בדיקה האם מכשיר שהוא לא מחשב (כמו טלפון לדוגמה) יכול להתחבר למערכת	הרצת שרט איזורי על מחשב והתחברות למכשיר עם טלפון	עובד, אך מכופה המערכת לא עובדת בצוואר אופטימלית עם טלפון (חוויות המשתמש אינה אופטימלית)

# מדריך למשתמש:

## פירוט כל קבצי המערכת:

static	4/25/2025 6:18 PM	File folder
templates	5/1/2025 11:02 PM	File folder
uploads	5/1/2025 10:48 PM	File folder
_init_.py	4/27/2025 12:21 AM	Python Source File 0 KB
app.py	5/1/2025 10:47 PM	Python Source File 4 KB
auth.py	5/1/2025 9:57 PM	Python Source File 5 KB
config.py	5/1/2025 10:00 PM	Python Source File 1 KB
database.db	5/1/2025 11:27 PM	Data Base File 28 KB
file_management.py	5/1/2025 11:18 PM	Python Source File 6 KB
file_management_routes.py	5/1/2025 11:09 PM	Python Source File 4 KB
friend_management.py	5/1/2025 11:18 PM	Python Source File 5 KB
friend_management_routes.py	5/1/2025 11:16 PM	Python Source File 5 KB
grand_server.py	5/1/2025 9:55 PM	Python Source File 7 KB
models.py	5/1/2025 9:48 PM	Python Source File 3 KB
server.crt	4/30/2025 4:39 PM	Security Certificate 2 KB
server.key	4/30/2025 4:39 PM	KEY File 2 KB
sync.py	5/2/2025 1:48 AM	Python Source File 7 KB

- app.py - הקובץ הראשי שמקים את המערכת (מקיים את האתר, מגדר את החיבור למסד הנתונים וכו'). מרייצ' בתוכו גם את sync.py.
- auth.py - אחראי על האוטנטיקציה של משתמשים (הרשמה והתחברות).
- config.py - מרכז את הקבועים וההגדרות של המערכת.
- database.db - מסד הנתונים של המערכת.
- file\_management.py - מכיל את המחלקה filemanager ואחראי על לוגיקת הקבצים.
- file\_management\_routes.py - ממפה את הפעולות על הקבצים ל-.http://.
- friend\_management.py - מכיל את המחלקה friendmanager ואחראי על לוגיקת החברויות.
- friend\_management\_routes.py - ממפה את הפעולות הקשורות לחברויות ל-.http://.
- grand\_server.py - השרת המרכזי האחראי על סyncron המערכת.
- models.py - הגדרת מודלים כגון user, file וכו'.
- server.crt - תעודת ssl של השרת.
- server.key - המפתח הציבורי של השרת.
- sync.py - אחראי על סינכרון השינויים עם השרת המרכזי.

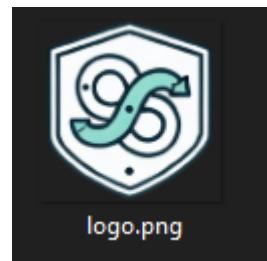
### **תוכן התקייה static:**

css	5/1/2025 11:19 PM	File folder
images	4/25/2025 6:18 PM	File folder

**תוכן התקייה CSS:**

 auth.css	5/1/2025 11:25 PM	CSS Source File	4 KB
 dashboard.css	5/1/2025 11:27 PM	CSS Source File	5 KB

- auth.css - קובץ css המופיע בדף התחברות וההרשמה.
- dashboard.css - קובץ css המופיע בשאר הדפים (אלה שאפשר לגשת אליהם רק לאחר התחברות).

**תוכן התקייה images:****תוכן התקייה templates:**

 dashboard.html	5/1/2025 11:22 PM	Chrome HTML Do...	4 KB
 friends_files.html	5/1/2025 3:47 PM	Chrome HTML Do...	2 KB
 friends_list.html	5/1/2025 10:56 PM	Chrome HTML Do...	3 KB
 incoming_requests.html	5/1/2025 11:02 PM	Chrome HTML Do...	3 KB
 login.html	5/1/2025 11:19 PM	Chrome HTML Do...	2 KB
 register.html	5/1/2025 11:19 PM	Chrome HTML Do...	2 KB

- דף html של מסך הבית. - dashboard.html
- דף html של מסך הקבצים של חבר. - friends\_files.html
- דף html של רשימת החברים. - friends\_list.html
- דף html של בקשות החברות. - incoming\_requests.html
- דף html של מסך ההתחברות. - login.html
- דף html של מסך ההרשמה. - register.html

**תוכן התקייה uploads הוא הקבצים המועלים למערכת. דוגמה:**

 3f8614b1efe5467cb6f0d2a38a01cd1f	5/1/2025 10:47 PM	File	297 KB
 cc2057b9db484cafae50405b40affca4.pdf	5/1/2025 10:47 PM	Microsoft Edge P...	88 KB

## התקנת המערכת:

על מנת להריץ את השירות, צריך לוודא את הדברים הבאים:

- מותקן pycharm visual studio code editor כגון code editor
- מותקן על המחשב python 3.12 ומעלה.
- הספריות הבאות מותקנות על המחשב:
  - flask ♦
  - flask-wtf ♦
  - SQLAlchemy ♦
  - Werkzeug ♦

ניתן להתקין את ספריות אלו באמצעות הרצת הפקודה בטרמינל:

- ```
pip install Flask Flask-WTF SQLAlchemy Werkzeug
```
- למחשב ישנו חיבור לאינטרנט.
  - בנוסף לכך, מומלץ לשנות קבועים מסוימים ב-`ukconfig` כגון המיקום של תיקיית הקבצים, הכתובת של השירות המרכזי, מקום תעודת ה-`ssl` והפתח הציבורי, סוג הקבצים המתאפשרים במערכת וכו'.

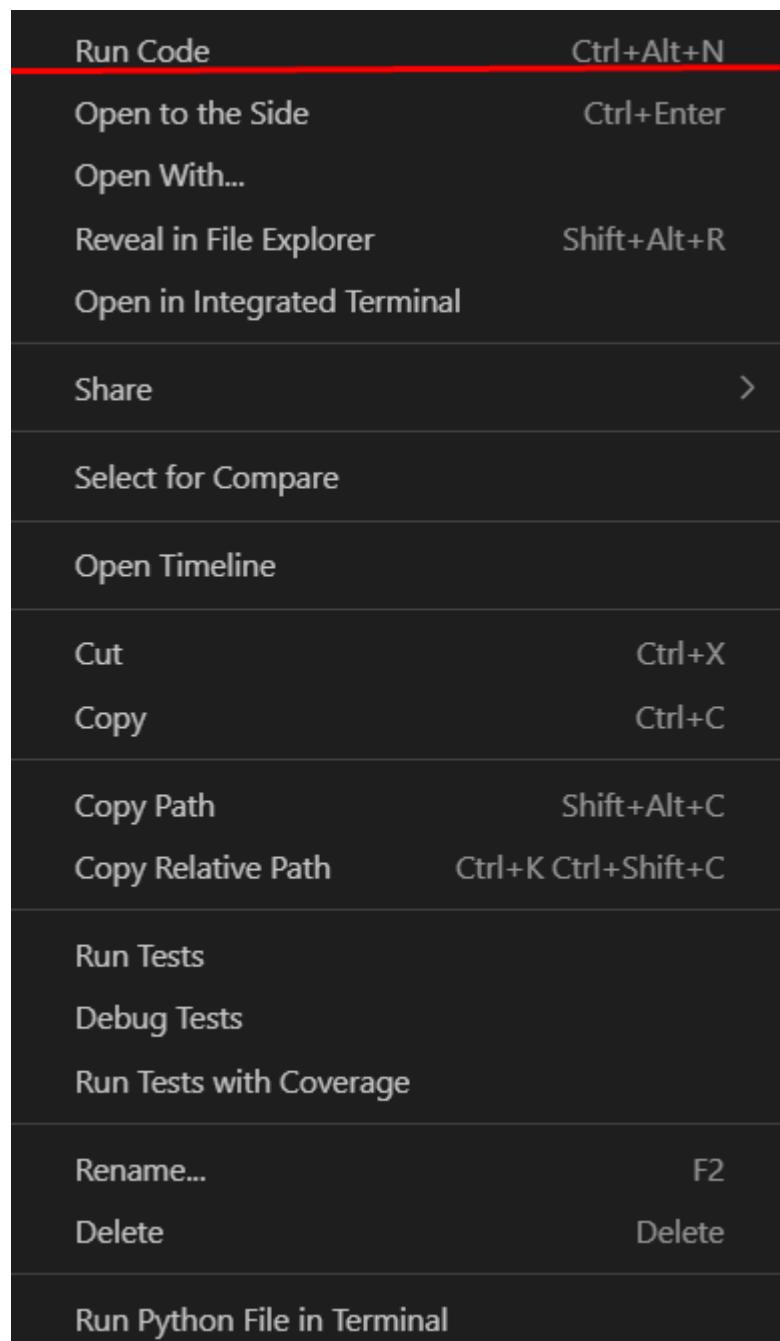
במידה ורוצים רק להתחבר למערכת (ולהסתמך על שירות אזורי שכבר רץ), הדרישות הללו לא רלוונטיות.

## הפעלת המערכת:

כעת, צריך לוודא שיש שירות מרכזי הפועל. במידה ואין, ניתן להריץ אותו כאשר מרים את הקובץ `grand_server.py`.

דרך visual studio code או terminal, ניתן להריץ את השירות באמצעות לחיצה ימנית על הקובץ ולחיצה על `run code` או כתיבת הפקודה ב-`terminal`:

```
python grand_server.py
```



במידה והכל רץ כשורה, יופיע ב-terminal:

```
[GrandServer] Listening on 0.0.0.0:9000
```

כעת, ניתן להריץ את אחד מן השירותים האזוריים. על מנת להריץ שירות זה, נדרש להריץ את הקובץ `app.py`. ניתן להריצו באמצעות אותו כפתור ב-visual studio code (יש ללחוץ קלאיק ימני על הקובץ `app.py` כעת) או להריץ את השורה ב-terminal:

```
python app.py
```

במידה והכל ה被执行 כשורה, יופיע ב-terminal:

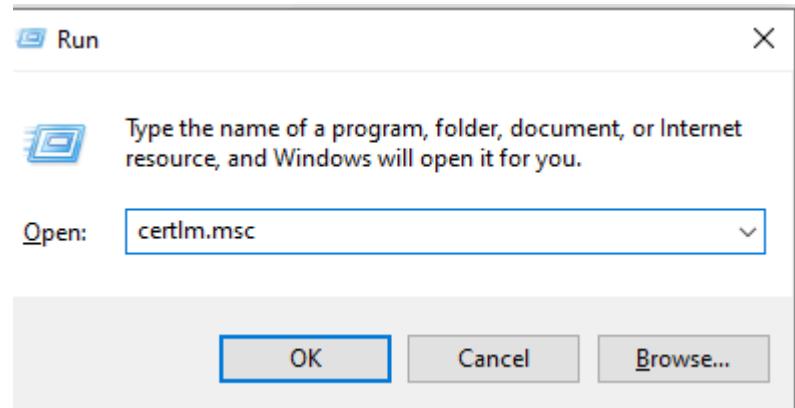
```
[Sync] Launching background sync thread...
* Serving Flask app 'app'
* Debug mode: off
[Sync] Connected to grand server
WARNING: This is a development server. Do not use it in a production deployment. Use Gunicorn or uWSGI instead.
* Running on all addresses (0.0.0.0)
* Running on https://127.0.0.1:5000
* Running on https://10.0.0.3:5000
Press CTRL+C to quit
```

וב-terminal של השרת המרכזי יופיע:

**[GrandServer] Regional connected (TLS): (10.0.0.3, 56279)**

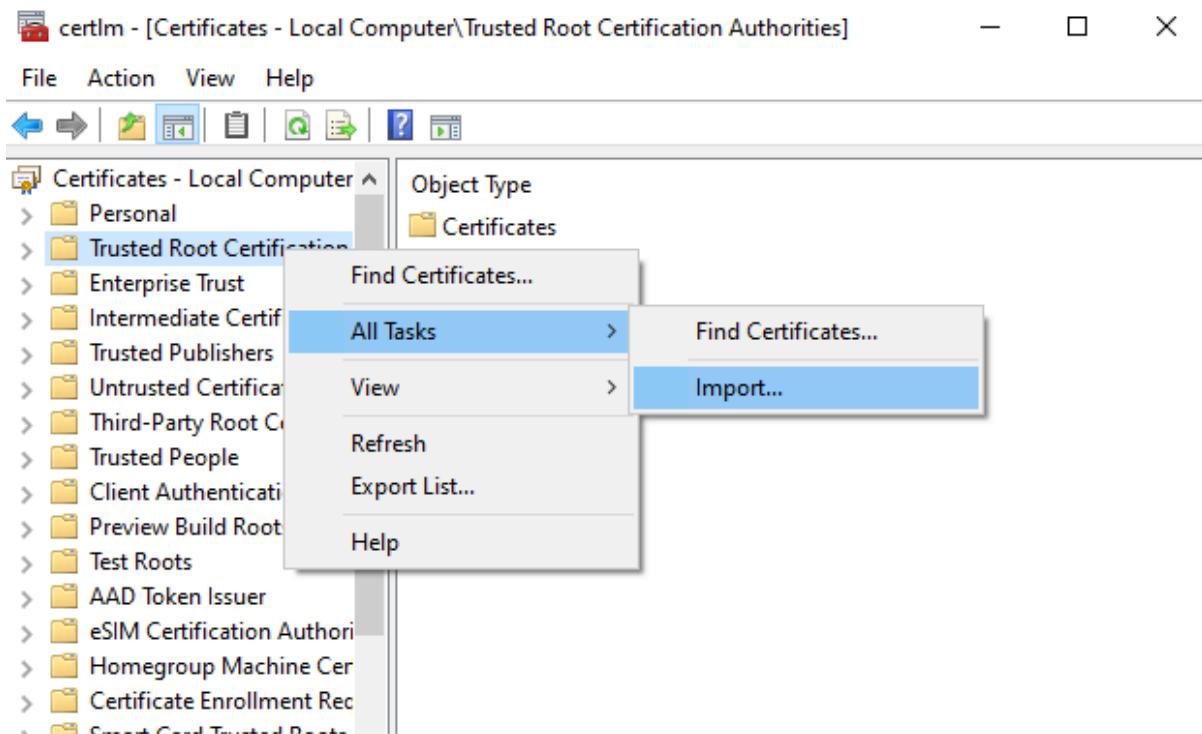
לאחר שביצעו כל השלבים האלו, ניתן להתחבר למערכת באמצעות הדפדפן. קודם כל, כיוון שתעודת ה-ssl של השירות אינה רשמית, צריך לוודא שהתעודה מוכרת על ידי המכשיר שמשתמשים בו. במידה ולשרת יש תעודה ssl רשמית, ניתן לדלג על החלק הבא:

על מנת להתחבר במכשיר מסויים צריך את תעודה ssl על המכשיר. ניתן לחוץ עכשו על + win R, בלחפש בלשונית החיפוש certlm.msc.

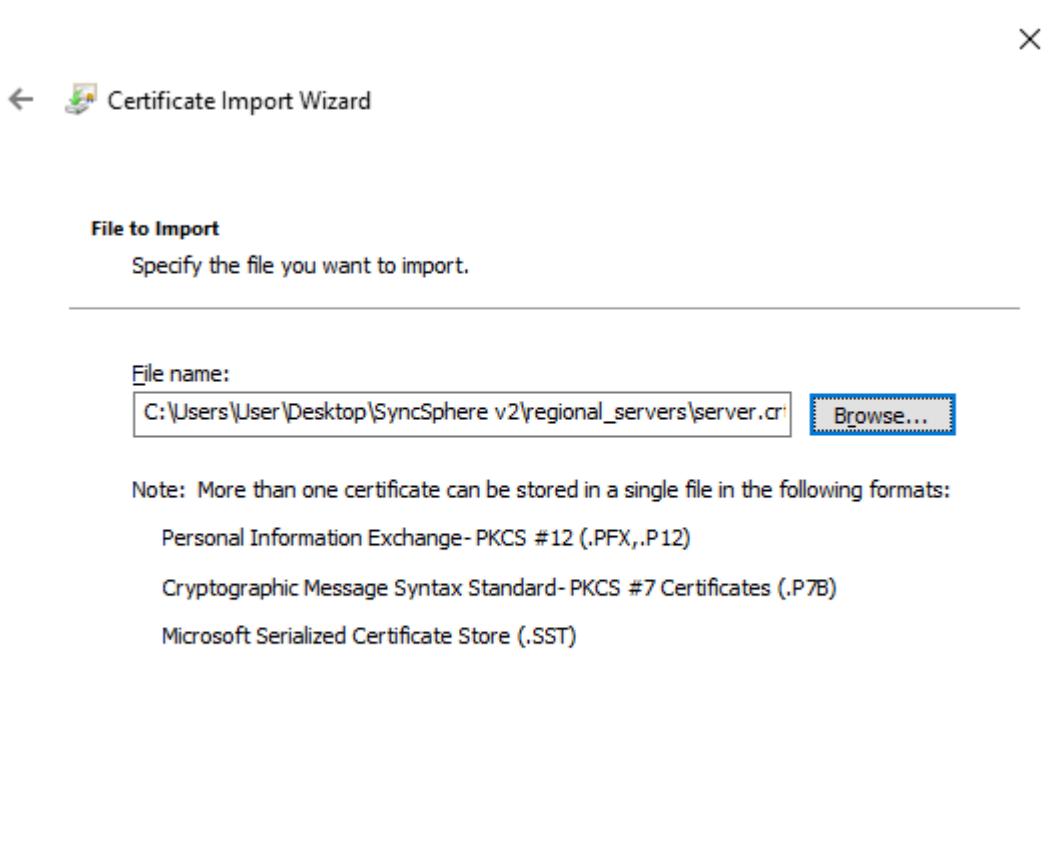


לאחר שלוחצים על הכption 'ok' ומאפשרים לתוכנה לבצע שינויים במחשב, לוחצים מקש ימני על ."Trusted root certification authorities", שמיים את העכבר על "all tasks" ולוחצים על ."import"

## SyncSphere



לאחר מכן, לוחצים על "next" ובלשונית החיפוש שמים את תעוזת ה-.iss.

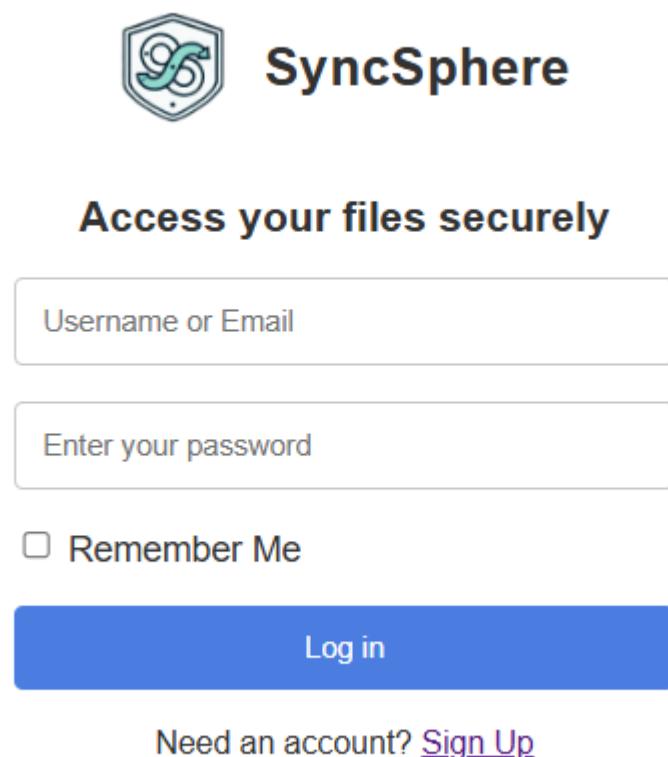


לאחר מכן, לוחצים פעמיים על "next" ולבסוף על "finish". סוגרים את כל החלונות שפתחנו וופתחים מחדש את הדף.

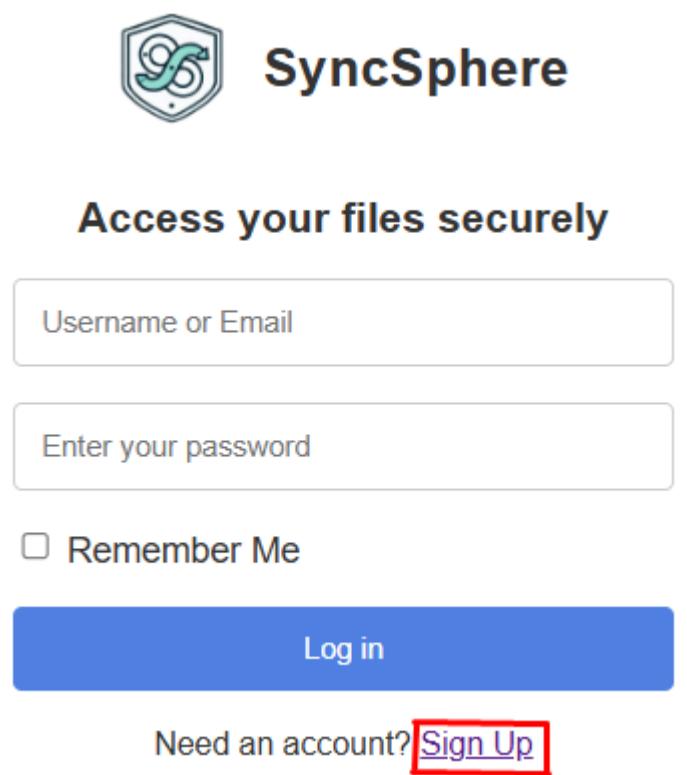
(מחלק זה ניתן להמשיך למי שיש תעודת **ssl רשמיית**):  
על מנת להתחבר, צריך לכתוב את ה-url הבא: `https://(server_ip):(server_port)` כאשר `server_ip` הוא כתובת ה-IP של השרת ו-`server_port` הוא הפורט שהשרת רץ עליו. דוגמה לכתובת כאשר IP השרת הוא 100.100.100.100 והפורט הוא 5000 יהיה:  
<https://100.100.100.100:5000>

### שימוש במערכת:

לאחר כניסה לכתובת ה-IP, יופיע העמוד הבא:

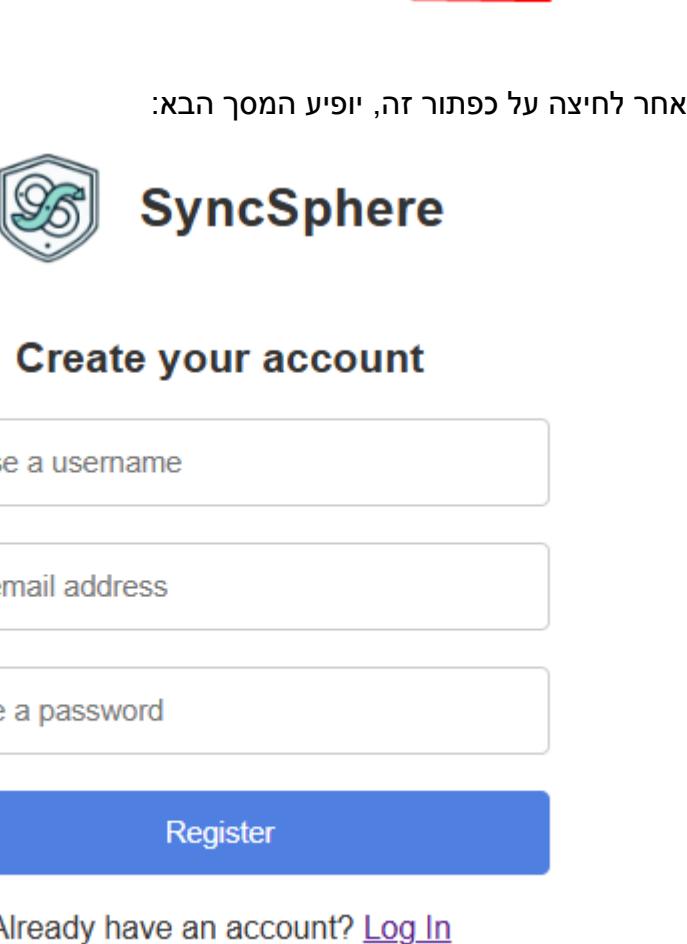


על מנת להשתמש במערכת, צריך משתמש פעיל. על מנת ליצור משתמש, ניתן ללחוץ על הכפתור "sign up".



The image shows the SyncSphere login page. At the top is the SyncSphere logo, which consists of a stylized 'S' inside a hexagonal frame. To the right of the logo is the word "SyncSphere". Below the logo is the text "Access your files securely". There are two input fields: one for "Username or Email" and one for "Enter your password". Below these fields is a checkbox labeled "Remember Me". A large blue button with the text "Log in" is centered below the password field. Below the login form, the text "Need an account? [Sign Up](#)" is displayed, with "Sign Up" being the link.

לאחר לחיצה על כפתור זה, יופיע המסר הבא:



The image shows the SyncSphere sign-up page. It features the same logo and header as the login page. Below the header is the text "Create your account". There are three input fields: "Choose a username", "Your email address", and "Create a password". A large blue button with the text "Register" is centered below the password field. Below the sign-up form, the text "Already have an account? [Log In](#)" is displayed, with "Log In" being the link.

עכשו, על מנת ליצור משתמש יש למלא את הפרטים בשדות המתאימים להן.

SyncSphere

במידה והפרטים תקניים (כטוב אימיל תקנית ויחודית, שם משתמש ייחודי), המשמש יוציא בהצלחה ומסך התחברות יקופץ בחזרה.icut, נותר למלא את הפרטים של המשתמש וללוחץ על כפתור ההתחברות.

במידה והתחברות עברית בהצלחה, המשתמש יופנה לדף הבא:



SyncSphere

Storage used: **0.0 GB** of **1.0 GB**

Dashboard Requests Friends List Logout

## Available Files

You have no uploaded files yet.

## Effortless File Management

Easily upload and organize your files with SyncSphere.

No file chosen

**למעלה בצד ימין, יש ארבעה כפתורים:**

- לחתורה למסך הבית dashboard
  - למעבר למסך בקשות החברות requests
  - למעבר למסך החברים friends list
  - להתנתקות מהמערכת logout

בצד העליון השמאלי של המסר, ניתן לראות את כמות האחסון שנוצלה במערכת ואת כמות האחסון המקסימלית.

The screenshot shows the SyncSphere dashboard. At the top, there's a navigation bar with the SyncSphere logo, 'Dashboard', 'Requests', 'Friends List', and 'Logout'. Below the navigation, a red-bordered box highlights 'Storage used: 0.0 GB of 1.0 GB'. The main area has two sections: 'Available Files' (with a message 'You have no uploaded files yet.') and 'Effortless File Management' (with a message 'Easily upload and organize your files with SyncSphere.' and a 'Choose File' button with 'No file chosen' and a green 'Upload File' button).

. "choose file" ללחוץ על הכפתור באמצעות המסך

This screenshot is similar to the first one, showing the SyncSphere dashboard. The 'choose file' button in the 'Effortless File Management' section is highlighted with a red border.

כעת, ניתן למצוא את הקובץ הרצוי במערכת הקבצים. לאחר שמוסאים את הקובץ, צריך ללחוץ על "upload file".

The screenshot shows the 'Effortless File Management' section of the SyncSphere dashboard. A file named 'test.txt' has been selected, as indicated by the text next to the 'Choose File' button. The 'Upload File' button is highlighted with a red border.

במידה והקובץ תקין (אין חריגה מכמות הדיכרון, סיוםת מאושרת בשורת האזרוי, קובץ קיימ) תופיע הودעה מתאימה והקובץ יופיע על המסך.

# SyncSphere

The screenshot shows the SyncSphere dashboard. At the top, there's a navigation bar with links for Dashboard, Requests, Friends List, and Logout. Below that, a message says "Storage used: 0.0 GB of 1.0 GB". A section titled "Available Files" lists a single file: "test.txt" uploaded on "2025-05-01 20:40" with size "0". Actions for this file include Download (blue button), Delete (red button), Private dropdown (set to "Private"), and Update (green button). Below this is a central area with the heading "Effortless File Management" and a sub-instruction "Easily upload and organize your files with SyncSphere." It includes a "Choose File" input field showing "No file chosen", an "Upload File" button, and a success message "File uploaded successfully!".

- cut, ניתן לראות את שם הקובץ, תאריך ההעלאה למערכת וגודל האחסון שהוא צריך.  
cut, ניתן לבצע מספר פעולות הקשורות לקובץ:  
• ניתן להוריד את הקובץ למכשיר בלחיצה על הכפתור "download":

## Available Files

| Filename | Upload Date      | Size (bytes) | Actions                                                                                            |
|----------|------------------|--------------|----------------------------------------------------------------------------------------------------|
| test.txt | 2025-05-01 20:40 | 0            | <button>Download</button> <button>Delete</button> <button>Private</button> <button>Update</button> |

- ניתן למחוק את הקובץ מן המערכת בלחיצה על הכפתור "delete":

## Available Files

| Filename | Upload Date      | Size (bytes) | Actions                                                                                            |
|----------|------------------|--------------|----------------------------------------------------------------------------------------------------|
| test.txt | 2025-05-01 20:40 | 0            | <button>Download</button> <button>Delete</button> <button>Private</button> <button>Update</button> |

- ניתן לשנות את הרשות הצפיה בקובץ בלשונית על הלשונית ליד הרשות הגישה, הגדרת הרשותה הרצiosa ואז הכפתור "update"

## Available Files

| Filename | Upload Date      | Size (bytes) | Actions                                                                                            |
|----------|------------------|--------------|----------------------------------------------------------------------------------------------------|
| test.txt | 2025-05-01 20:40 | 0            | <button>Download</button> <button>Delete</button> <button>Private</button> <button>Update</button> |

## SyncSphere

במידה ורוצים להויספ' חבר למערכת, ניתן לעשות זאת באמצעות מעבר למסך בקשות החברות:

The screenshot shows the SyncSphere application interface. At the top, there's a navigation bar with a logo, the text "SyncSphere", and links for "Dashboard", "Requests" (which is highlighted with a red box), "Friends List", and "Logout". Below the navigation, a message says "Space used: 0.0 GB of 1.0 GB".

**Available Files**

| Filename | Upload Date      | Size (bytes) | Actions                                                                                       |
|----------|------------------|--------------|-----------------------------------------------------------------------------------------------|
| test.txt | 2025-05-02 20:40 | 0            | <a href="#">Download</a> <a href="#">Delete</a> <a href="#">Public</a> <a href="#">Update</a> |

**Effortless File Management**

Easily upload and organize your files with SyncSphere.

[Choose File](#) | No file chosen [Upload File](#)

כעת, המסך הבא יופיע:

The screenshot shows the "Friend Requests" section of the SyncSphere application. At the top, there's a navigation bar with a logo, the text "SyncSphere", and links for "Dashboard", "Friends List", and "Logout".

**Friend Requests**

Add a friend [Send Request](#)

No pending requests.

על מנת להויספ' חבר, כתובים את שמו בלשונית החיפוש ולוחצים על הכפתור "send request":

The screenshot shows the "Friend Requests" section again. A user named "friend" has pending requests. The "Send Request" button for this user is highlighted with a red box.

**Friend Requests**

friend [Send Request](#)

No pending requests.

במידה והבקשה תקינה, תופיע על כך הודעה וביקשת החברות תופיע על המסך:

## Friend Requests

Add a friend Send Request

**Your Pending Requests**

**test**  
@test

No pending requests.

Friend request sent!

בקשות חברות שנשלחו למשתמש יופיעו גם הן במסך זה. במידה ורוצים לאשר/לדוחות בקשה, יש להליץ על הceptor המתאים לכך:

## Friend Requests

Add a friend Send Request

**Incoming Requests**

**test**  
@test

Accept Reject

## SyncSphere

כעת, כדי לראות את החברים של המשתמש, ניתן לעבור למסך החברים המופיע בפינה הימנית למעלה בדף (ניתן לעبور לדף זה גם מדף הבית בלבד על אותו כפתור):

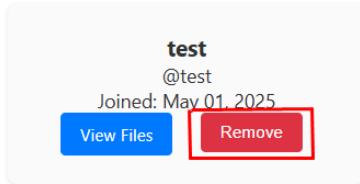
The screenshot shows the 'Friends List' page. At the top, there is a navigation bar with icons for user profile, SyncSphere logo, Dashboard, Friends List (which is highlighted with a red box), and Logout. Below the navigation bar, the main content area has a title 'Friend Requests' and a button 'Add a friend'. A message 'No pending requests.' is displayed. A green notification box contains the text 'Request accepted.'.

כעת, בדף זה יופיעו כל החברים של המשתמש:

The screenshot shows the 'Your Friends' page. At the top, there is a navigation bar with icons for user profile, SyncSphere logo, Dashboard, Requests, and Logout. Below the navigation bar, the main content area has a title 'Your Friends' and a card for a friend named 'test'. The card displays the friend's name, handle (@test), and joining date (Joined: May 01, 2025). It also includes two buttons: 'View Files' and 'Remove'.

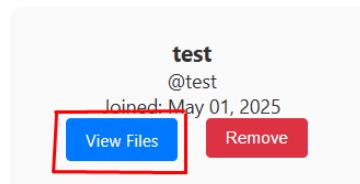
על מנת להסיר חבר במידת הצורך, ניתן ללחוץ על כפתור ה-'remove' המופיע ליד שם החבר.

## Your Friends



על מנת לצפות בקבצים של חבר, ניתן ללחוץ על הכפתור "view files".

## Your Friends



לאחר לחיצה על כפתור זה, יופיעו קבצי המשתמש, נתונים הקשורים אליום ואפשרות להורדה.

| Filename | Upload Date      | Size | Actions                  |
|----------|------------------|------|--------------------------|
| test.txt | 2025-05-01 20:40 | 0    | <a href="#">Download</a> |

שים לב - ניתן לראות רק את הקבצים שהרשאות הגישה שלהם היא "public".

במידה ורוצים להוריד את הקובץ, ניתן ללחוץ על הכפתור "download":

### test's Files

| Filename | Upload Date      | Size | Actions                  |
|----------|------------------|------|--------------------------|
| test.txt | 2025-05-01 20:40 | 0    | <a href="#">Download</a> |

במידה ורוצים להתנתק מהמערכת, ניתן ללחוץ על הכפתור "logout" המופיע בפינה הימנית העליונה של כל עמוד:

The screenshot shows the SyncSphere application interface. At the top, there is a navigation bar with the SyncSphere logo, the text "SyncSphere", and three links: "Dashboard", "Friends", and "Logout". The "Logout" link is highlighted with a red box. Below the navigation bar, there is a section titled "test's Files" which contains a table with one row of data. The table has four columns: "Filename", "Upload Date", "Size", and "Actions". The data in the table is as follows:

| Filename | Upload Date      | Size | Actions                  |
|----------|------------------|------|--------------------------|
| test.txt | 2025-05-01 20:40 | 0    | <a href="#">Download</a> |

**כעת אפשר להשתמש במערכת ולתפועל אותה בפשטות!**

## **סיכום אישי - רפלקציה:**

אני אתחילה מנוסה, תהlixir העבודה על הפרויקט היה מהנה ומאתגר בו זמן. אתחילה מ釐ין שהזה הפעם הראשונה שאני עובד על פרויקט בסדר גודל זהה, אך שתחילה לא ידעתן כיצד לגשת לכל התהlixir. תחילתה רצית להקדיש זמן רב לתוכנן המערכת והפרויקט בכלל, על מנת למנוע בזבוז זמן מיותר בעבודה על הפרויקט. עם זאת, אני חשב שתוכנן הפרויקט שלי לך בחסר. לא ידעתן באמת להעיר בזורה טובעה פרויקט בסדר גודל זהה, אך שלא הקדשתי זמן רב לדברים שבפועל לך לו זמן גדול. לדוגמה, בתכנון הפרויקט הקדשתי הרבה פחות זמן ממה שלקח בפועל לעצב את משק המשתמש ואת דפי האינטראקט. דזוקא לחלק זה כנ' התחררתי, ומצאתי את עצמי הרבה פעמים לא מסתפק בעיצוב מסוים לדף שלא מצא חן בעיני. במהלך פרויקט זה נדרש ממוני המן למידה עצמית בתחוםים שונים שפחות הכרתוי. הבולטים בהם היו שימוש בספריה flask ובעיצוב דפי אינטראקט באמצעות CSS ו-HTML. אני מרגיש שהסתיבת המרכזית לכך שהצלחתו ללמידה את נושא זה הייתה היוצאות בחברים המכירים כבר את הנושא (חלקים עבדו על עיצוב אתרים בכיתה י' לדוגמה) וכך מכך ולא הבנתי משהו, הם הרגישו חופשי לעזרות לי ולהסביר במידת הצורך. וכך מכך, אני מרגיש שננהנית מהלמידה העצמית, גם אם הוא קשיים מסוימים בחלוקת מהנושאים (כגון עבודה עם מסד הנתונים). בסה"כ, אני מרגיש שהפרויקט תרם לי רבות לפיתוח יכולות למידה עצמית וחקיר באינטרנט. דבר ממשמעוני שהורד לי מוטיבציה במהלך העבודה על הפרויקט היה שאיפת יתר. בתכנון המערכת היו פיצרים מסוימים שרציתי לכלול חוץ מהחוונים שקייםים במערכת, אך את חלקם לא הצלחתה/הספיקתי לעבוד עליהם בצורה מספקת טובה על מנת להסביר אותם. במהלך כתיבת הקוד עצמו, היו חלקים שייתר נהנית ופחות נהנית. אישית נהנית ממחשבה על האלגוריתם של סנכiron השרטטים השונים וננהנית מלחשוב על אופטימיזציות שונות שיכלתי לבצע בקשר לאלגוריתם זה. תהlixir כתיבת תיק הפרויקט גם היה לי קשה בהתחלה, שכן כמו הפרויקט עצמו אין לי ניסיון רב בעיצוב תיקי פרויקט בסדר גודל זהה. חלק זה דזוקא פחות נהנית שכן אני מרגיש שתהlixir הכתיבה חוזר על עצמו לעיתים קרובות והרגשתי שחוורתה על דברים שכתבתי בחלוקת מוקדים יותר של הפרויקט מספר פעמים. בראייה לאחר, אני חשב שהיית מתכוון את הזמן של יותר בזירות והיית מקצה יותר זמן ממה שאני חשב שהיה לוקח לי למקורה והיית נתקל בבעיות בדרך (כמו שקרה בפועל). אחד מן הכליםocy החשובים שאני לוקח מהפרויקט הוא התמדה במסהו אחד למשך זמן. בעבר כשעבדתי על פרויקטים שעבודות שונות מספר פעמים הרבה עלי כהה פרויקטים במקביל/הפרויקטם לקחו כמות זמן קצרה יחסית כך שאף פעם באמת לא יצא לי לעבוד על מהו מסוים למשך כל כך הרבה זמן. בנוסף לכך, אני מרגיש שיכילות הלמידה העצמית והתוכנו לטווח ארוך של השתרפו עקב העבודה בפרויקט. במידה מסוימת אני מרגיש שכן היה לי את הקשרים הללו ברמה מסוימת, אך עקב הגדל של הפרויקט איתגרתי את עצמי גם בתחוםים אלה. במבט לאחר, אני חשב שהיית מתכוון לוגיקת המערכת לפני העבודה על משק המשתמש וכו'. אני אישית חשב שלוגיקת המערכת הוא החלק החשוב באמת והיית צריך להסתכל על עיצוב ממשק המשמש כמעין פרס או בונוס שיכלתי לבצע במקום לשים את זה כפוקוס מרכזי.

לסיכום, העבודה על הפרויקט הייתה בהחלט תהlixir מיוחד וחידש בשביי, ולמרות כל הקשיים שהו בדרך (והיו קשיים), אני מאמין שאזכור את העבודה על הפרויקט כחויה חיובית בסך הכל.

אני רוצה להודות למורה שלי צביקה שליווה אותי למשך כל התהlixir ולחברי לכיתה שהוא חשובים אליו ועזרו לי במידת הצורך בכל בקשה שהיא הקשורה לפרויקט ומהוצאה לו.

## **ביבליוגרפיה:**

GeeksforGeeks. (2023, April 24). *Flask Tutorial*. GeeksforGeeks.

<https://www.geeksforgeeks.org/flask-tutorial/>

*SQLAlchemy Documentation — SQLAlchemy 2.0 Documentation*. (n.d.).

Docs.sqlalchemy.org. <https://docs.sqlalchemy.org/en/20/>

*ssl — TLS/SSL wrapper for socket objects — Python 3.7.2 documentation*. (2018). Python.org. <https://docs.python.org/3/library/ssl.html>

SuperSimpleDev. (2022, February 5). *HTML & CSS Full Course - Beginner to Pro (2022)*. Wwww.youtube.com.

<https://www.youtube.com/watch?v=G3e-cpL7ofc>

**נוכחות:****קוד הפרוייקט:**: app.py

```

import os
import threading
from datetime import timedelta

from flask import Flask, render_template, session,
               redirect, url_for, flash
from flask_wtf import CSRFProtect

from models import db
from auth import auth_bp
from file_management_routes import files_bp
from friend_management_routes import friends_bp
from config import certfile, keyfile, basedir,
                  db_file, secret_key
from sync import connect_to_server

(__app = Flask(__name__))

Secret key for signing session cookies and CSRF #
tokens
app.config['SECRET_KEY'] = secret_key

(SQLite database URI (absolute path to file #
app.config['SQLALCHEMY_DATABASE_URI'] =
      "{f"sqlite:/// {db_file

```

```

app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] =
    False # disable event system to save memory

        } ) app.config.update

Only send session cookie over HTTPS #
    ,SESSION_COOKIE_SECURE': True'
Prevent JavaScript access to the cookie #
    ,SESSION_COOKIE_HTTPONLY': True'
Mitigate CSRF by restricting cross-site #
                                sending of cookies
    ,SESSION_COOKIE_SAMESITE': 'Lax'
Lifetime of a "permanent" session #
    PERMANENT_SESSION_LIFETIME': '
                                (timedelta(days=7
                                ({

Initialize the SQLAlchemy extension #
        (db.init_app(app

Wrap the app in Flask-WTF's CSRFProtect to #
                                auto-validate tokens on all POST forms
        (csrf = CSRFProtect(app

        .../Authentication routes: /auth #
            app.register_blueprint(auth_bp,
                                ('url_prefix='/auth
        .../File management routes: /files #
            app.register_blueprint(files_bp,
                                ('url_prefix='/files
        .../Friend management routes: /friends #

```

```
app.register_blueprint(friends_bp,
                      url_prefix='/friends')

# routes
@app.route('/')
def dashboard():
    Ensure we have a user_id in session (1 #
    ('uid = session.get('user_id'
                         :if not uid
    ('return redirect(url_for('auth.login

Lazy imports to avoid circular dependencies #
from models import User
from file_management import FileManager

Load current user from DB using (2 #
Session.get() to avoid legacy warning
(user = db.session.get(User, uid
                         :if user is None
Stale or invalid session → clear and #
                           force re-login
    ()session.clear
("flash("Please log in again.", "error"
('return redirect(url_for('auth.login

Instantiate FileManager and fetch the (3 #
                           user's files
file_manager =
FileManager(upload_folder=os.path.join(basedir,
   ''uploads
(files = file_manager.list_user_files(user
```

```
Render the dashboard template with user (4 #
   data
   ) return render_template
   , 'dashboard.html'
   , username=user.username
   , files=files
   , current_user=user
   (
   : '__if __name__ == '__main__
Create database tables if they don't exist #
   : ()with app.app_context
   ()db.create_all

Launch the background sync thread exactly #
   once
   print("[Sync] Launching background sync
   (thread...", flush=True
   threading.Thread(target=connect_to_server,
   ()daemon=True).start

Start the Flask HTTPS server (self-signed #
   (cert for dev
   )app.run
   , 'host='0.0.0.0
   debug=False, # disable
   debugger in production
   (ssl_context=(certfile, keyfile
   (
```

**auth.py:**

```
from flask import Blueprint, render_template,
request, redirect, url_for, flash, session
from werkzeug.security import
generate_password_hash, check_password_hash
from sqlalchemy import or_
from models import db, User
from config import changes_queue
from datetime import datetime, timedelta

# Create a Blueprint for all auth-related routes
(register, login, logout)
auth_bp = Blueprint('auth', __name__)

# Lockout policy: max failed attempts before
# temporary block
MAX_LOGIN_ATTEMPTS = 5
LOCKOUT_TIME_MINUTES = 15

def _is_locked_out():
    """ Check if the current session is under
lockout. Returns True if lockout hasn't expired;
otherwise clears lockout state."""
    lockout_until = session.get('lockout_until')
    if lockout_until and
datetime.fromisoformat(lockout_until) >
datetime.now():
        return True
```

```
# Lockout expired or not set → remove any
stale counters
    session.pop('lockout_until', None)
    session.pop('login_attempts', None)
    return False

@auth_bp.route('/register', methods=['GET',
'POST'])
def register():
    """ Handle user registration."""
    if request.method == 'POST':
        session.clear() # Prevent session
fixation attacks

        username =
request.form['username'].strip()
        email      = request.form['email'].strip()
        password = request.form['password']

        # Basic input validation
        if not username or not email or not
password:
            flash('Please fill in all fields!', 'error')
            return
redirect(url_for('auth.register'))

        # Check for existing user by username OR
email (safe ORM filter, avoids SQLi)
        existing = User.query.filter(
```

```
        or_(User.username == username,
User.email == email)
    ).first()
    if existing:
        flash('Username or Email already
exists!', 'error')
        return
    redirect(url_for('auth.register'))

    # Create and hash the new user's password
    new_user = User(username=username,
email=email)
    new_user.set_password(password)
    db.session.add(new_user)
    db.session.commit()

    # Notify other regional servers of the new
user
    changes_queue.put({
        "type": "user_create",
        "user_id": new_user.id,
        "username": new_user.username,
        "email": new_user.email,
        "password": password,
        "timestamp":
datetime.now().isoformat()
    })

    flash('Registration successful! Please log
in.', 'success')
```

```
        return redirect(url_for('auth.login'))\n\n    # GET request → render form template\n    return render_template('register.html')\n\n\n@auth_bp.route('/login', methods=['GET', 'POST'])\ndef login():\n    """ Handle user login. """\n    if _is_locked_out():\n        flash('Too many failed attempts. Try again later.', 'error')\n        # Render the login template without\n        # redirect to preserve lockout message\n        return render_template('login.html')\n\n    if request.method == 'POST':\n        session.clear() # Prevent session\n        fixation on every login attempt\n\n        identifier =\n        request.form['username_or_email'].strip()\n        password = request.form['password']\n        remember = 'remember_me' in request.form\n\n    # Checkbox presence\n\n        # Lookup by username OR email using safe\n        # ORM parameter binding\n        user = User.query.filter(\n
```

```
        or_(User.username == identifier,
User.email == identifier)
    ).first()

        # Verify password using Werkzeug's
constant-time check

        if user and user.check_password(password):
            # Reset failed-attempt counters on
success

            session.pop('login_attempts', None)
            session.pop('lockout_until', None)
            # Store minimal user info in session
            session['user_id'] = user.id
            session['username'] = user.username
            session.permanent = remember # honor "Remember Me"

            return redirect(url_for('dashboard'))

        # Failed login: increment counter
        attempts = session.get('login_attempts',
0) + 1
        session['login_attempts'] = attempts

        # If too many fails, impose lockout window
        if attempts >= MAX_LOGIN_ATTEMPTS:
            until = datetime.now() +
timedelta(minutes=LOCKOUT_TIME_MINUTES)
            session['lockout_until'] =
until.isoformat()
```

```

        flash(f'Too many failed attempts. Try
again in {LOCKOUT_TIME_MINUTES} minutes.',

'error')
    else:
        flash('Invalid credentials, please try
again.', 'error')

    return redirect(url_for('auth.login'))

# GET request → render the login form
return render_template('login.html')

@auth_bp.route('/logout')
def logout():
    """ Log the user out by clearing the session
and redirecting to login."""
    session.clear()
    flash('Logged out successfully!', 'success')
    return redirect(url_for('auth.login'))

```

**config.py:**

```

import os
from queue import Queue

# Hostname or IP of the Grand Server
GRAND_HOST = "10.0.0.3"
GRAND_PORT = 9000

BIND_HOST = "0.0.0.0"

```

```

basedir =
os.path.abspath(os.path.dirname(__file__)) #The
location of the system.
certfile = os.path.join(basedir, 'server.crt')
#The location of the certificate.
keyfile = os.path.join(basedir, 'server.key') #The
location of the public key
secret_key =
"dfcc0ab22a9913f9f19c758feabe1c2c56d0e5be670647ef4
f131666c36f0572" #The secret key of the project.

db_file = os.path.join(basedir, 'database.db')
UPLOAD_FOLDER = os.path.join(os.getcwd(),
'uploads')

changes_queue = Queue()

ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg',
'jpeg', 'gif'}

```

**file\_management\_routes.py:**

```

from flask import Blueprint, request, flash,
redirect, url_for, send_from_directory, session
from models import User
from file_management import FileManager
import os

```

```
# Blueprint for file operations, all routes under
/files
files_bp = Blueprint('files', __name__)

# Determine absolute path to the uploads directory
basedir      =
os.path.abspath(os.path.dirname(__file__))
upload_dir = os.path.join(basedir, 'uploads')

# Instantiate FileManager with the upload folder
path
file_manager =
FileManager(upload_folder=upload_dir)

def get_current_user():
    """ Helper to retrieve the logged-in User from
session. Returns None if no valid user_id in
session. """
    user_id = session.get('user_id')
    return User.query.get(user_id) if user_id else
None

@files_bp.route('/upload', methods=['POST'])
def upload():
    """ Handle file upload: """
    user = get_current_user()
    if not user:
        flash("Please log in first.", "error")
```

```
        return redirect(url_for('auth.login'))\n\n    # Get the FileStorage object from the form\n    file_storage = request.files.get('file')\n    if not file_storage or not\n        file_storage.filename:\n        flash("No file selected for upload.",\n              "error")\n        return redirect(url_for('dashboard'))\n\n    try:\n        # Attempt to save file; may flash quota or\n        type errors internally\n        file_manager.save_file(file_storage, user)\n        flash("File uploaded successfully!",\n              "success")\n    except ValueError as e:\n        # Known validation error from FileManager\n        flash(str(e), "error")\n    except Exception:\n        # Unexpected error\n        flash("An unexpected error occurred while\n        uploading.", "error")\n\n    return redirect(url_for('dashboard'))\n\n@files_bp.route('/download/<int:file_id>')\ndef download(file_id):
```

```
""" Serve a file download if the user has
permission. """
user = get_current_user()
file_record =
file_manager.get_file_record(file_id)
if not file_record:
    flash("File not found.", "error")
    return redirect(url_for('dashboard'))

# Ensure user owns the file or it's public
if not
file_manager.is_access_allowed(file_record, user):
    flash("Access not allowed.", "error")
    return redirect(url_for('dashboard'))

# Send the stored file under its original
filename
return send_from_directory(
    file_manager.upload_folder,
    file_record.stored_filename,
    as_attachment=True,
download_name=file_record.original_filename
)

@files_bp.route('/delete/<int:file_id>',
methods=['POST'])
def delete(file_id):
```

```
""" Delete a file record and its physical
file. """
    user = get_current_user()
    file_record =
file_manager.get_file_record(file_id)
    if not file_record:
        flash("File not found.", "error")
        return redirect(url_for('dashboard'))

try:
    file_manager.delete_file(file_record,
user)
    flash("File deleted successfully.",
"success")

except Exception as e:
    flash(str(e), "error")

return redirect(url_for('dashboard'))


@files_bp.route('/permissions/<int:file_id>',
methods=['POST'])
def change_permissions(file_id):
    """Update a file's access permissions (private
or public). """
    user = get_current_user()
    if not user:
        flash("Please log in first.", "error")
        return redirect(url_for('auth.login'))
```

```
file_record =
file_manager.get_file_record(file_id)
if not file_record:
    flash("File not found.", "error")
    return redirect(url_for('dashboard'))

# Only the owner can change permissions
if file_record.user_id != user.id:
    flash("Access not allowed.", "error")
    return redirect(url_for('dashboard'))

new_perm = request.form.get('permissions')
try:

file_manager.update_permissions(file_record,
new_perm, user)
    flash("Permissions updated!", "success")
except Exception as e:
    flash(str(e), "error")

return redirect(url_for('dashboard'))
```

### file\_management.py:

```
import os
import uuid
from werkzeug.utils import secure_filename
from models import db, File
from config import changes_queue
from datetime import datetime
from config import ALLOWED_EXTENSIONS
```

```
class FileManager:  
    #Encapsulates file operations.  
    def __init__(self, upload_folder):  
        """ Initialize FileManager with a  
        directory to store uploaded files. Creates the  
        directory if it doesn't exist. """  
        self.upload_folder = upload_folder  
        os.makedirs(self.upload_folder,  
exist_ok=True)  
  
    def allowed_file(self, filename):  
        """ Check if the file has an allowed  
        extension. Returns True if extension is in  
        ALLOWED_EXTENSIONS. """  
        return '.' in filename and  
filename.rsplit('.', 1)[1].lower() in  
ALLOWED_EXTENSIONS  
  
    def generate_unique_filename(self, filename):  
        """ Generate a unique filename using UUID4  
        and preserve the file extension. """  
        ext = filename.rsplit('.', 1)[1] if '.' in  
filename else ''  
        # Combine a random hex string with the  
        # original extension  
        unique_name = f"{uuid.uuid4().hex}.{ext}"  
if ext else uuid.uuid4().hex  
        return unique_name
```

```
def save_file(self, file_storage, user):
    """ Save an uploaded file to disk and
create a corresponding DB record. Also updates the
user's used_storage and enqueues a sync event. """
    # Validate file presence
    if not file_storage or
file_storage.filename == '':
        raise ValueError("No file provided.")
    # Validate extension
    if not
self.allowed_file(file_storage.filename):
        raise ValueError("File type not
allowed.")

    # Secure the original filename and
generate a unique stored name
    original_filename =
secure_filename(file_storage.filename)
    unique_filename =
self.generate_unique_filename(original_filename)
    file_path =
os.path.join(self.upload_folder, unique_filename)

    # Save to disk and measure file size
    file_storage.save(file_path)
    file_size = os.path.getsize(file_path)

    # Enforce user storage quota
    if user.used_storage + file_size >
user.storage_quota:
```

```
os.remove(file_path)
raise ValueError("Storage quota
exceeded.")

# Create DB record for the file
file_record = File(
    user_id=user.id,
    stored_filename=unique_filename,
    original_filename=original_filename,
    file_size=file_size,
    permissions='private'
)
db.session.add(file_record)

# Update user's used storage and commit
both changes
user.used_storage += file_size
db.session.add(user)
db.session.commit()

# Read file content for sync event
with open(file_path, 'rb') as f:
    content = f.read()

# Enqueue a file_upload event for
synchronization
changes_queue.put({
    "type": "file_upload",
    "payload": {
```

```
        "id":  
file_record.id,  
        "user_id":  
file_record.user_id,  
        "stored_filename":  
file_record.stored_filename,  
        "original_filename":  
file_record.original_filename,  
        "upload_date":  
file_record.upload_date.isoformat(),  
        "file_size":  
file_record.file_size,  
        "permissions":  
file_record.permissions,  
        "content": content  
    },  
    "timestamp":  
datetime.now().isoformat()  
}  
  
return file_record  
  
def list_user_files(self, user):  
    """ Return a list of File records  
belonging to the given user. """  
    return  
File.query.filter_by(user_id=user.id).all()  
  
def get_file_record(self, file_id):
```

```
    """ Retrieve a single File record by its
ID. Returns None if not found.

    """
    return File.query.get(file_id)

    def delete_file(self, file_record, user,
enqueue=True):
        """
        Delete a file from disk and remove its
DB record. Only the file owner may delete.
Enqueues a file_delete sync event.
        """
        if file_record.user_id != user.id:
            # Prevent unauthorized deletions
            raise PermissionError("You are not
authorized to delete this file.")

            # Remove file from disk
            file_path =
os.path.join(self.upload_folder,
file_record.stored_filename)
            if os.path.exists(file_path):
                os.remove(file_path)
            # Remove DB record
            db.session.delete(file_record)
            db.session.commit()

            # Notify other servers of deletion
            if enqueue:
                changes_queue.put({
                    "type": "file_delete",
                    "file_id": file_record.id,
```

```
        "timestamp":  
    datetime.now().isoformat()  
    })  
    return True  
  
    def update_permissions(self, file_record,  
new_permissions, user, enqueue=True):  
        """ Change the permission of a file  
(private or public) Only the owner may change  
permissions. Enqueues a permission_change sync  
event. """  
        if file_record.user_id != user.id:  
            # Prevent unauthorized permission  
changes  
            raise PermissionError("You are not  
authorized to change permissions for this file.")  
        file_record.permissions = new_permissions  
        db.session.commit()  
  
        # Notify other servers of permission  
change  
        if enqueue:  
            changes_queue.put({  
                "type":  
"permission_change",  
                "file_id": file_record.id,  
                "new_permissions":  
file_record.permissions,  
                "timestamp":  
datetime.now().isoformat()
```

```
        } )

    return file_record

    def is_access_allowed(self, file_record,
user):
        """ Check if the given user may access the
file. Owners always allowed, otherwise only public
files. """
        if file_record.user_id == user.id:
            return True
        return file_record.permissions == 'public'
```

**friend\_managment\_routes.py:**

```
from flask import Blueprint, render_template,
request, redirect, url_for, flash, session
from models import User
from friend_management import FriendManager
from file_management import FileManager

# Blueprint for friend-related routes under
/friends
friends_bp = Blueprint('friends', __name__)
# Manager instances for business logic
friend_manager = FriendManager()
file_manager =
FileManager(upload_folder='uploads')

def get_current_user():
```

```
""" Retrieve the logged-in user from the
session. Returns None if no user is logged in. """
user_id = session.get('user_id')
return User.query.get(user_id) if user_id else
None

@friends_bp.route('/requests', methods=['GET',
'POST'])
def view_requests():
    """ GET: Show incoming & outgoing friend
requests, and a form to send new ones. POST:
Handle submission of a new friend request by
username. """
    user = get_current_user()
    if not user:
        return redirect(url_for('auth.login'))

    # Handle new request submission
    if request.method == 'POST' and 'to_username' in request.form:
        to_username = request.form['to_username']
        to_user =
User.query.filter_by(username=to_username).first()
        if not to_user:
            flash("User not found.", "error")
        else:
            try:
                friend_manager.send_request(user,
to_user)
```

```
        flash("Friend request sent!",  
"success")  
    except ValueError as e:  
        flash(str(e), "error")  
    return  
redirect(url_for('friends.view_requests'))  
  
# Build lists for template  
raw_in =  
friend_manager.get_incoming_requests(user)  
incoming = [  
    {'req': fr, 'sender':  
User.query.get(fr.from_user_id)}  
    for fr in raw_in  
]  
  
raw_out =  
friend_manager.get_outgoing_requests(user)  
outgoing = [ User.query.get(fr.to_user_id) for  
fr in raw_out ]  
  
return render_template(  
    'incoming_requests.html',  
    incoming=incoming,  
    outgoing=outgoing  
)  
  
@friends_bp.route('/requests/respond/<int:rq_id>',  
methods=['POST'])
```

```
def respond_request(rq_id):
    """ Handle acceptance or rejection of a friend
request. """
    user = get_current_user()
    action = request.form.get('action')
    try:
        friend_manager.respond_request(rq_id,
accept=(action == 'accept'))
        flash(f"Request {action}ed.", "success")
    except ValueError as e:
        flash(str(e), "error")
    return
redirect(url_for('friends.view_requests'))


@friends_bp.route('/list')
def list_friends():
    """ Display the current user's friend list.
"""
    user = get_current_user()
    if not user:
        return redirect(url_for('auth.login'))
    friends = friend_manager.get_friends(user)
    return render_template('friends_list.html',
friends=friends)


@friends_bp.route('/remove/<username>',
methods=['POST'])
def remove(username):
```

```
""" Remove an existing friendship. """
user = get_current_user()
if not user:
    return redirect(url_for('auth.login'))

to_user =
User.query.filter_by(username=username).first()
if not to_user:
    flash("User not found.", "error")
else:
    try:
        friend_manager.remove_friend(user,
to_user)
        flash("Friend removed.", "success")
    except ValueError as e:
        flash(str(e), "error")

return
redirect(url_for('friends.list_friends'))


@friends_bp.route('/<username>/files')
def view_friend_files(username):
    """ View files shared by a specific friend.
Only files with 'public' permission or owned by
the friend are shown.
"""

user = get_current_user()
if not user:
    return redirect(url_for('auth.login'))
```

```
friend =
User.query.filter_by(username=username).first()
if not friend:
    flash("User not found.", "error")
    return
redirect(url_for('friends.list_friends'))


friends_list =
friend_manager.get_friends(user)
if friend not in friends_list:
    flash("You may only view files of your
friends.", "error")
    return redirect(url_for('dashboard'))


all_files      =
file_manager.list_user_files(friend)
# Filter out files the current user isn't
allowed to see
allowed_files = [f for f in all_files if
file_manager.is_access_allowed(f, user)]


return render_template(
    'friends_files.html',
    files=allowed_files,
    friend=friend
)
```

**friend\_management.py:**

```
from models import db, User, FriendRequest,
Friendship
from config import changes_queue
from datetime import datetime

class FriendManager:
    #Encapsulates friend request and friendship
operations, including sending requests,
responding, listing, and removal.

    def send_request(self, from_user, to_user,
enqueue=True):
        """ Send a friend request from 'from_user'
to 'to_user'. """
        # Prevent sending a request to oneself
        if from_user.id == to_user.id:
            raise ValueError("Cannot friend
yourself.")

        # Check existing pending request in either
direction
        existing = FriendRequest.query.filter_by(
            from_user_id=from_user.id,
            to_user_id=to_user.id,
            status='pending'
        ).first()
        if not existing:
            existing =
FriendRequest.query.filter_by(
```

```
        from_user_id=to_user.id,
        to_user_id=from_user.id,
        status='pending'
    ).first()
if existing:
    raise ValueError("Friend request
already pending.")

# Check if users are already friends
fri = Friendship.query.filter_by(
    user_id=from_user.id,
    friend_id=to_user.id
).first()
if not fri:
    fri = Friendship.query.filter_by(
        user_id=to_user.id,
        friend_id=from_user.id
    ).first()
if fri:
    raise ValueError("You are already
friends.")

# Create and persist the new friend
request
fr = FriendRequest(
    from_user_id=from_user.id,
    to_user_id=to_user.id
)
db.session.add(fr)
db.session.commit()
```

```
# Enqueue change for synchronization with
other nodes

    if enqueue:
        changes_queue.put({
            "type": "friend_request",
            "request_id": fr.id,
            "from_user": fr.from_user_id,
            "to_user": fr.to_user_id,
            "timestamp":
datetime.now().isoformat()
        })
    return fr

def get_incoming_requests(self, user):
    """ Retrieve all pending friend requests
where 'user' is the recipient. """
    return FriendRequest.query.filter_by(
        to_user_id=user.id,
        status='pending'
    ).all()

def get_outgoing_requests(self, user):
    """ Retrieve all pending friend requests
sent by 'user'. """
    return FriendRequest.query.filter_by(
        from_user_id=user.id,
        status='pending'
    ).all()
```

```
def respond_request(self, request_id,
accept=True, enqueue=True):
    """ Accept or reject a friend request by
ID. If accepted, creates reciprocal Friendship
entries. """
    # Fetch the request object
    fr = FriendRequest.query.get(request_id)
    if not fr:
        raise ValueError("Friend request not
found.")
    # Update status
    fr.status = 'accepted' if accept else
'rejected'
    if accept:
        # Create mutual friendship records
        db.session.add(Friendship(user_id=fr.from_user_id,
friend_id=fr.to_user_id))
        db.session.add(Friendship(user_id=fr.to_user_id,
friend_id=fr.from_user_id))
        db.session.commit()

    # Enqueue corresponding sync event
    if enqueue:
        if accept:
            changes_queue.put({
                "type": "friend_added",
                "request_id": fr.id,
```

```
        "timestamp":  
    datetime.now().isoformat()  
        } )  
    else:  
        changes_queue.put({  
            "type":  
"friend_rejected",  
            "request_id": fr.id,  
            "from_user": fr.from_user_id,  
            "to_user": fr.to_user_id,  
            "timestamp":  
    datetime.now().isoformat()  
        } )  
    return fr  
  
def get_friends(self, user):  
    """ List all users who are friends with  
'user'. """  
        # Query one direction, then load User for  
each friendship  
    fs =  
Friendship.query.filter_by(user_id=user.id).all()  
        return [User.query.get(f.friend_id) for f  
in fs]  
  
def remove_friend(self, user, to_user,  
enqueue=True):  
    """ Remove an existing friendship between  
'user' and 'to_user'. """  
        # Find both directions of the friendship
```

```
f1 =
Friendship.query.filter_by(user_id=user.id,
friend_id=to_user.id).first()
f2 =
Friendship.query.filter_by(user_id=to_user.id,
friend_id=user.id    ).first()
if not f1 and not f2:
    raise ValueError("Friendship not
found.")

# Delete whichever records exist
if f1:
    db.session.delete(f1)
if f2:
    db.session.delete(f2)
db.session.commit()

# Enqueue removal event
if enqueue:
    changes_queue.put({
        "type":      "friend_removed",
        "user_id":   user.id,
        "friend_id": to_user.id,
        "timestamp": datetime.now().isoformat()
    })
return True
```

## [models.py:](#)

```
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
from werkzeug.security import
generate_password_hash, check_password_hash

db = SQLAlchemy()

class User(db.Model):
    #A class that represents a user in the system.
    id = db.Column(db.Integer, primary_key=True)
    #The id of the user.

    username = db.Column(db.String(80),
unique=True, nullable=False) #The username.

    email = db.Column(db.String(120), unique=True,
nullable=False) #The email of the user.

    password_hash = db.Column(db.String(128),
nullable=False) #The hashed password of the user.

    created_at = db.Column(db.DateTime,
default=datetime.now) #The creation date of the
user.

    used_storage = db.Column(db.Integer,
nullable=False, default=0) #The used storage of
the user.

    storage_quota = db.Column(db.Integer,
nullable=False, default=1073741824) #The max
storage of the user.

    def set_password(self, password):
```

```
        """Generating the hash value of the
password."""
        self.password_hash =
generate_password_hash(password)

    def check_password(self, password):
        """Checking if the password is the same as
the user's password."""
        return
check_password_hash(self.password_hash, password)

class File(db.Model):
    #Represents the metadata of a file in the
    system.
    id = db.Column(db.Integer, primary_key=True)
    #The id of the file
    user_id = db.Column(db.Integer,
db.ForeignKey('user.id'), nullable=False) #The id
    of the file's owner.
    stored_filename = db.Column(db.String(128),
nullable=False) #The name of the file in the file
    system.
    original_filename = db.Column(db.String(128),
nullable=False) #The original name of the file.
    upload_date = db.Column(db.DateTime,
default=datetime.now) #The creation date of the
    file.
    file_size = db.Column(db.Integer) #The file's
size.
```

```
permissions = db.Column(db.String(32),
default='private') #The file's viewing permission.

class FriendRequest(db.Model):
    #Represents a friend request in the system.
    id = db.Column(db.Integer, primary_key=True)
    #The id of the request.
    from_user_id = db.Column(db.Integer,
db.ForeignKey('user.id'), nullable=False) #The id
of the sender.
    to_user_id = db.Column(db.Integer,
db.ForeignKey('user.id'), nullable=False) #The id
of the receiver.
    status = db.Column(db.String(16),
default='pending') #The status of the friend
request('pending', 'accepted', 'rejected')
    created_at = db.Column(db.DateTime,
default=datetime.now) #The creation date of the
friend request.

class Friendship(db.Model):
    #Represents a friendship in the system.
    id = db.Column(db.Integer, primary_key=True)
    #The id of the friendship.
    user_id = db.Column(db.Integer,
db.ForeignKey('user.id'), nullable=False) #The id
of one of the user.
    friend_id = db.Column(db.Integer,
db.ForeignKey('user.id'), nullable=False) #The id
of the other user.
```

```
    created_at = db.Column(db.DateTime,
default=datetime.now) #The creation date of the
friendship.
```

## sync.py:

```
import os
import socket
import json
import base64
import time
import ssl
from datetime import datetime
from queue import Empty

from config import GRAND_HOST, GRAND_PORT,
changes_queue, UPLOAD_FOLDER

def send_changes(sock):
    """ Drain the local changes_queue, encode file
contents in Base64 when needed, and send a single
'changes' packet to the Grand Server over the
socket."""
    events = []
    # Pull all pending events without blocking
    while True:
        try:
            change = changes_queue.get_nowait()
        except Empty:
```

```
        break

        # If it's a new file upload, Base64-encode
        raw bytes for JSON transport
        if change['type'] == 'file_upload':
            payload_copy =
change['payload'].copy()
            raw = payload_copy.pop('content')    #
remove binary data from payload
            # encode bytes to UTF-8 string so it
can be embedded in JSON
            payload_copy['content'] =
base64.b64encode(raw).decode('utf-8')
            events.append({
                'type': change['type'],
                'payload': payload_copy,
                'timestamp': change['timestamp']
            })
        else:
            # other event types can be sent as-is
            events.append(change)

        # If no events to send, do nothing
if not events:
    return

        # Build and send the JSON packet, ending with
newline for framing
packet = {'type': 'changes', 'events': events}
```

```
sock.sendall((json.dumps(packet) +  
\n').encode('utf-8'))  
  
def receive_changes(message):  
    """ Apply incoming events from Grand Server  
inside a fresh Flask application context. This  
allows us to modify the database and file system  
safely."""  
    from app import app  
    with app.app_context():  
        from file_management import FileManager  
        from friend_management import  
FriendManager  
        from models import db, User, File  
  
        file_manager =  
FileManager(upload_folder=UPLOAD_FOLDER)  
        friend_manager = FriendManager()  
  
        for sync_event in message.get('events',  
[]):  
            event_type = sync_event.get('type')  
            try:  
                if event_type == 'file_upload':  
                    # Decode and write file bytes  
                    payload =  
sync_event['payload']  
                    data =  
base64.b64decode(payload['content'])
```

```
        dest =
os.path.join(UPLOAD_FOLDER,
payload['stored_filename'])

os.makedirs(os.path.dirname(dest), exist_ok=True)
with open(dest, 'wb') as f:
    f.write(data)

# Merge into DB (insert or
update existing record by ID)
rec = File(
    id=payload['id'],

user_id=payload['user_id'],
stored_filename=payload['stored_filename'],
original_filename=payload['original_filename'],
file_size=payload['file_size'],
permissions=payload['permissions'],

upload_date=datetime.fromisoformat(payload['upload_date'])
)
db.session.merge(rec)
db.session.commit()

elif event_type == 'file_delete':
```

```
        rec =
file_manager.get_file_record(sync_event['file_id'])
)
        if not rec:
            print(f"[Sync] Warn: no
file record for deletion ID
{sync_event['file_id']}]", flush=True)
            continue
        user =
User.query.get(rec.user_id)
        file_manager.delete_file(rec,
user, enqueue=False)

        elif event_type ==
'permission_change':
        rec =
file_manager.get_file_record(sync_event['file_id'])
)
        if not rec:
            print(f"[Sync] Warn: no
file for permission change ID
{sync_event['file_id']}]", flush=True)
            continue
        user =
User.query.get(rec.user_id)

        file_manager.update_permissions(rec,
sync_event['new_permissions'], user,
enqueue=False)
```

```
        elif event_type == 'user_create':
            # Merge new user record,
preserves existing if present
            u = User(
                id=sync_event['user_id'],
                username=sync_event['username'],
                email=sync_event['email']
            )

            u.set_password(sync_event['password'])
            db.session.merge(u)
            db.session.commit()

        elif event_type ==
'friend_request':
            friend_manager.send_request(
User.query.get(sync_event['from_user']),
User.query.get(sync_event['to_user']),
enqueue=False
)

        elif event_type == 'friend_added':
            friend_manager.respond_request(sync_event['request_id'],
accept=True, enqueue=False)
```

```
        elif event_type ==  
        'friend_rejected':  
  
            friend_manager.respond_request(sync_event['request_id'], accept=False, enqueue=False)  
  
        elif event_type ==  
        'friend_removed':  
            friend_manager.remove_friend(  
                User.query.get(sync_event['user_id']),  
  
                User.query.get(sync_event['friend_id']),  
                enqueue=False  
            )  
  
        else:  
            # Unknown event type - log for  
            # debugging  
            print(f"[Sync] Unknown event  
type: {event_type}", flush=True)  
  
    except Exception as e:  
        # Roll back any partial DB changes  
        # on error  
        db.session.rollback()  
        print(f"[Sync] Error applying  
'{event_type}' event: {e}", flush=True)
```

## SyncSphere

```
def sync_changes(sock):
    """ Read newline-delimited JSON commands from Grand Server and dispatch to send_changes or receive_changes handlers."""
    buffer = sock.makefile('r') # wrap socket in file-like object for line reads
    for line in buffer:
        try:
            received_message = json.loads(line)
        except json.JSONDecodeError:
            continue

        message_type =
received_message.get('type')
        if message_type == 'send':
            send_changes(sock)
        elif message_type == 'receive':
            receive_changes(received_message)

        time.sleep(0) # yield to other threads

    # When connection closes, clean up
    sock.close()
    print("[Sync] Connection closed", flush=True)

def connect_to_server():
    """ Establish a TLS-wrapped TCP connection to the Grand Server, then enter the sync loop."""

```

```
# Create SSL context for client with no
verification (dev only)
context =
ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
context.check_hostname = False
context.verify_mode = ssl.CERT_NONE

raw_sock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
sock = context.wrap_socket(raw_sock,
server_hostname=GRAND_HOST)

sock.connect((GRAND_HOST, GRAND_PORT))
print("[Sync] Connected to grand server",
flush=True)
sync_changes(sock)
```

## grand\_server.py:

```
import socket
import threading
import json
import time
import ssl

from config import BIND_HOST, GRAND_PORT,
certfile, keyfile
```

## SyncSphere

```
# How often (in seconds) to prompt regionals for
changes
SYNC_INTERVAL = 30 # 1 sync every 30 seconds

# We want to keep all batches from the last 5 sync
intervals (5×SYNC_INTERVAL seconds)
HISTORY_WINDOW = SYNC_INTERVAL * 5

# history holds dicts {"ts": timestamp, "events":
[...]}
history = []

# Global list of connected regional sockets
clients = []
clients_lock = threading.Lock() # ensure one
thread at a time touches clients list

def send_history(conn):
    """
        Replay every batch of events from the last
HISTORY_WINDOW seconds
        to this newly connected regional_server.
    """
    cutoff = time.time() - HISTORY_WINDOW
    # send each batch whose timestamp is within
the window
    for entry in history:
        if entry["ts"] < cutoff:
            continue
```

```
packet = json.dumps({'type': 'receive',
'events': entry["events"]}) + '\n'
try:
    conn.sendall(packet.encode('utf-8'))
except Exception as e:
    print(f"[GrandServer] Failed to replay history to {conn.getpeername()}: {e}", flush=True)

def accept_loop(server_sock, context):
    """Continuously accept new regional connections and wrap them with TLS."""
    while True:
        try:
            raw_conn, addr = server_sock.accept()
            # Perform TLS handshake on the new connection
            try:
                conn =
context.wrap_socket(raw_conn, server_side=True)
                print(f"[GrandServer] Regional connected (TLS): {addr}", flush=True)
            except ssl.SSLError as e:
                print(f"[GrandServer] SSL handshake failed with {addr}: {e}", flush=True)
                raw_conn.close()
                continue
            # add to clients under lock
            with clients_lock:
```

```
        clients.append(conn)

        # replay missed-history batches
        send_history(conn)

        # start handler thread

threading.Thread(target=client_handler,
args=(conn,), daemon=True).start()

    except Exception as e:
        print(f"[GrandServer] accept_loop
error: {e}", flush=True)
        time.sleep(1)

def client_handler(conn):
    """Read incoming 'changes' messages and
rebroadcast them."""
    addr = conn.getpeername()
    f = conn.makefile('r')  # treat socket as file
for line-based reading
    try:
        for line in f:
            line = line.strip()
            if not line:
                continue

    try:
        msg = json.loads(line)
```

```
        except json.JSONDecodeError as e:
            print(f"[GrandServer] Invalid JSON
from {addr}: {e}", flush=True)
            continue

        mtype = msg.get('type')
        if mtype == 'changes':
            events = msg.get('events', [])
            print(f"[GrandServer] Received
{len(events)} events from {addr}", flush=True)
            broadcast(events, exclude=conn)
        else:
            print(f"[GrandServer] Unknown
message type from {addr}: {mtype}", flush=True)

    except Exception as e:
        print(f"[GrandServer] Connection error
from {addr}: {e}", flush=True)
    finally:
        # remove disconnected client
        with clients_lock:
            if conn in clients:
                clients.remove(conn)
    try:
        conn.close()
    except:
        pass
    print(f"[GrandServer] Regional
disconnected: {addr}", flush=True)
```

```
def sync_loop():
    """Every SYNC_INTERVAL seconds, send 'send' to
all live regionals."""
    while True:
        try:
            time.sleep(SYNC_INTERVAL)
            print("[GrandServer] Requesting
changes from all regionals...", flush=True)
            packet = json.dumps({'type': 'send'})
+ '\n'
            data = packet.encode('utf-8')

            with clients_lock:
                for conn in list(clients):
                    try:
                        conn.sendall(data)
                    except Exception as e:
                        addr = None
                        try:
                            addr =
conn.getpeername()
                        except:
                            pass
                        print(f"[GrandServer]
Error sending sync request to {addr}: {e}",
flush=True)
                        clients.remove(conn)
                        try:
                            conn.close()

```

```
        except:
            pass

        except Exception as e:
            print(f"[GrandServer] sync_loop error: {e}", flush=True)
            time.sleep(1)

def broadcast(events, exclude=None):
    """
    Broadcast received events to every regional
    except the sender,
    and record each batch with a timestamp for
    history-window replay.

    """
    # record this batch with the current time
    history.append({
        "ts": time.time(),
        "events": events
    })
    # purge any entries older than HISTORY_WINDOW
    cutoff = time.time() - HISTORY_WINDOW
    history[:] = [h for h in history if h["ts"] >= cutoff]

    packet = json.dumps({'type': 'receive',
    'events': events}) + '\n'
    data = packet.encode('utf-8')
```

```
with clients_lock:
    for conn in list(clients):
        if conn is exclude:
            continue
        try:
            conn.sendall(data)
        except Exception as e:
            addr = None
            try:
                addr = conn.getpeername()
            except:
                pass
            print(f"[GrandServer] Broadcast error to {addr}: {e}", flush=True)
            clients.remove(conn)
            try:
                conn.close()
            except:
                pass

def main():
    # Create TCP listening socket
    server_sock = socket.socket(socket.AF_INET,
                                socket.SOCK_STREAM)
    # allow quick reuse after restart
    server_sock.setsockopt(socket.SOL_SOCKET,
                           socket.SO_REUSEADDR, 1)
    server_sock.bind((BIND_HOST, GRAND_PORT))
    server_sock.listen()
```

```
    print(f"[GrandServer] Listening on\n{BIND_HOST}:{GRAND_PORT}", flush=True)

    # Create TLS context and load certificate/key
    context =
        ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
        context.load_cert_chain(certfile=certfile,
keyfile=keyfile)

    # start accept and sync loops in background
    threads
        threading.Thread(target=accept_loop,
args=(server_sock, context), daemon=True).start()
        threading.Thread(target=sync_loop,
daemon=True).start()

    try:
        # keep main thread alive
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        print("[GrandServer] Shutting down.",
flush=True)
    finally:
        server_sock.close()

if __name__ == '__main__':
    main()
```

**templates/dashboard.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SyncSphere - Dashboard</title>
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/dashboard.css') }}>
</head>
<body>
    <header class="navbar">
        <div class="navbar-left">
            <a href="{{ url_for('dashboard') }}" class="logo-link" style="display:flex; align-items:center; text-decoration:none;">
                
                <span class="app-name">SyncSphere</span>
            </a>
        </div>
        <nav class="navbar-right">
            <a href="{{ url_for('dashboard') }}>Dashboard</a>
            <a href="{{ url_for('friends.view_requests') }}>Requests</a>
```

```
<a href="{{ url_for('friends.list_friends') } }">Friends List</a>
<a href="{{ url_for('auth.logout') } }">Logout</a>
</nav>
</header>

<section class="storage-usage">
    {%
        set used_gb = (current_user.used_storage /
        (1024*1024*1024)) %}
    {%
        set quota_gb = (current_user.storage_quota /
        (1024*1024*1024)) %}
    <p>Storage used: <strong>{{ used_gb|round(2) }} GB</strong> of <strong>{{ quota_gb|round(2) }} GB</strong></p>
</section>

<main>
    <section class="files-section">
        <h2>Available Files</h2>
        {% if files %}
            <table class="files-table">
                <thead>
                    <tr>
                        <th>Filename</th>
                        <th>Upload Date</th>
                        <th>Size (bytes)</th>
                        <th>Actions</th>
                    </tr>
                </thead>
```

```
<tbody>
    {%
        for file in files %
    <tr>
        <td>{{ file.original_filename }}</td>
        <td>{{ file.upload_date.strftime("%Y-%m-%d %H:%M") }}</td>
        <td>{{ file.file_size }}</td>
        <td>
            <a href="#"{{ url_for('files.download', file_id=file.id) }}>Download</a>
            <form action="#"{{ url_for('files.delete', file_id=file.id) }} method="post" style="display:inline;">
                <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
                <button type="submit" class="btn btn-delete" onclick="return confirm('Are you sure?')">Delete</button>
            </form>
            <form action="#"{{ url_for('files.change_permissions', file_id=file.id) }} method="post" style="display:inline; margin-left:0.5rem;">
                <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
                <select name="permissions" class="permission-select">
```

```
<option value="private" {%
if file.permissions=='private' %}selected{%
%}>Private</option>
<option value="public" {%
if file.permissions=='public' %}selected{%
%}>Public</option>
</select>
<button type="submit"
class="btn btn-upload">Update</button>
</form>
</td>
</tr>
{%
endfor %
}
</tbody>
</table>
{%
else %
<p class="no-files">You have no uploaded
files yet.</p>
{%
endif %
}
</section>

<section class="upload-cta">
<h2>Effortless File Management</h2>
<p>Easily upload and organize your files
with SyncSphere.</p>
<form action="{{ url_for('files.upload') }}"%
method="post" enctype="multipart/form-data"
class="upload-form">
<input type="hidden" name="csrf_token"
value="{{ csrf_token() }}">
```

```
<input type="file" name="file" required>
<button type="submit" class="btn
btn-upload">Upload File</button>
</form>
</section>

{%
  with messages =
get_flashed_messages(with_categories=True) %}
  {% if messages %}
    <div class="flash-messages">
      <ul>
        {% for category, message in messages %}
          <li class="{{ category }}>{{ message }}</li>
        {% endfor %}
      </ul>
    </div>
  {% endif %}
  {% endwith %}
</main>
</body>
</html>
```

### **templates/friends\_files.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

## SyncSphere

```
<title>{{ friend.username }}'s Files</title>
<meta name="viewport"
content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="{{ url_for('static', filename='css/dashboard.css') }}>
</head>
<body>
<header class="navbar">
<div class="navbar-left">
<a href="{{ url_for('dashboard') }}"
class="logo-link" style="display:flex;
align-items:center; text-decoration:none;">

alt="SyncSphere Logo" class="logo">
<span class="app-name">SyncSphere</span>
</a>
</div>
<nav class="navbar-right">
<a href="{{ url_for('dashboard') }}>Dashboard</a>
<a href="{{ url_for('friends.list_friends') }}>Friends</a>
<a href="{{ url_for('auth.logout') }}>Logout</a>
</nav>
</header>

<main class="main-content">
```

```
<section class="files-section">
    <h2>{{ friend.display_name or
friend.username }}'s Files</h2>
    {% if files %}
        <table class="files-table">
            <thead>
                <tr>
                    <th>Filename</th>
                    <th>Upload Date</th>
                    <th>Size</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                {% for file in files %}
                    <tr>
                        <td>{{ file.original_filename
} }</td>
                        <td>{{ file.upload_date.strftime("%Y-%m-%d %H:%M") }}</td>
                        <td>{{ file.file_size }}</td>
                        <td><a href="{{ url_for('files.download', file_id=file.id) }}"
class="btn btn-download">Download</a></td>
                    </tr>
                {% endfor %}
            </tbody>
        </table>
    {% else %}

```

```
<p class="no-files">No files  
available.</p>  
    {  
        % endif  
    }  
  </section>  
  </main>  
</body>  
</html>
```

### **templates/friends\_list.html:**

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="UTF-8">  
    <title>SyncSphere - Your Friends</title>  
    <meta name="viewport"  
content="width=device-width, initial-scale=1.0">  
    <link rel="stylesheet" href="{{ url_for('static', filename='css/dashboard.css') }}>  
  </head>  
  <body>  
    <header class="navbar">  
      <div class="navbar-left">  
        <a href="{{ url_for('dashboard') }}"  
class="logo-link" style="display:flex;  
align-items:center; text-decoration:none;">  
          
```

## SyncSphere

```
<span class="app-name">SyncSphere</span>
</a>
</div>
<nav class="navbar-right">
    <a href="{{ url_for('dashboard') }}>Dashboard</a>
    <a href="{{ url_for('friends.view_requests') }}>Requests</a>
    <a href="{{ url_for('auth.logout') }}>Logout</a>
</nav>
</header>

<main class="main-content">
    <section class="list-section">
        <h2>Your Friends</h2>

        {% if friends %}
            <div class="friends-grid">
                {% for friend in friends %}
                    <div class="friend-card">
                        <div class="friend-info">
                            <h3>{{ friend.display_name or friend.username }}</h3>
                            <p>@{{ friend.username }}</p>
                            <p>Joined: {{ friend.created_at.strftime('%b %d, %Y') }}</p>
                        </div>
                        <div class="friend-actions">
```

```
<a href="#">  
url_for('friends.view_friend_files',  
username=friend.username) } }"  
class="btn btn-download">View  
Files</a>  
  
<form action="#">  
url_for('friends.remove',  
username=friend.username) } }"  
method="post"  
style="display:inline;  
margin-left:0.5rem;">  
  
    <input type="hidden"  
name="csrf_token" value="#">  
    <button type="submit"  
class="btn btn-remove"  
onclick="return  
confirm('Remove friend?')">  
        Remove  
    </button>  
</form>  
</div>  
</div>  
{ % endfor %}  
</div>  
{% else %}  
    <p class="no-friends">You have no friends  
yet.</p>  
{ % endif %}
```

```
{% with messages =
get_flashed_messages(with_categories=True) %}
{% if messages %}
<div class="flash-messages">
<ul>
    {% for category, message in messages %}
        <li class="{{ category }}>{{ message }}</li>
    {% endfor %}
</ul>
</div>
{% endif %}
{% endwith %}
</section>
</main>
</body>
</html>
```

### **templates/incoming\_requests.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SyncSphere - Friend Requests</title>
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
```

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/dashboard.css') }}>
</head>
<body>
    <header class="navbar">
        <div class="navbar-left">
            <a href="{{ url_for('dashboard') }}" class="logo-link" style="display:flex; align-items:center; text-decoration:none;">
                
                <span class="app-name">SyncSphere</span>
            </a>
        </div>
        <nav class="navbar-right">
            <a href="{{ url_for('dashboard') }}>Dashboard</a>
            <a href="{{ url_for('friends.list_friends') }}>Friends List</a>
            <a href="{{ url_for('auth.logout') }}>Logout</a>
        </nav>
    </header>

    <main class="main-content">
        <section class="list-section">
            <h2>Friend Requests</h2>
        </section>
    </main>
</body>
```

```
<form method="post" action="{{ url_for('friends.view_requests') }}"  
class="action-form">  
    <input type="hidden" name="csrf_token"  
value="{{ csrf_token() }}">  
    <input type="text" name="to_username"  
placeholder="Add a friend" required>  
    <button type="submit" class="btn  
btn-upload">Send Request</button>  
</form>  
  
{% if outgoing %}  
    <h3>Your Pending Requests</h3>  
    <div class="friends-grid">  
        {% for u in outgoing %}  
            <div class="friend-card">  
                <div class="friend-info">  
                    <h3>{{ u.display_name or  
u.username }}</h3>  
                    <p>@{{ u.username }}</p>  
                </div>  
            </div>  
        {% endfor %}  
    </div>  
{% endif %}  
  
{% if incoming %}  
    <h3>Incoming Requests</h3>  
    <div class="friends-grid">  
        {% for item in incoming %}
```

```
<div class="friend-card">
    <div class="friend-info">
        <h3>{{ item.sender.display_name or
item.sender.username }}</h3>
        <p>@{{ item.sender.username }}</p>
    </div>
    <div class="friend-actions">
        <form method="post" action="{{
url_for('friends.respond_request',
rq_id=item.req.id) }}">
            <input type="hidden"
name="csrf_token" value="{{ csrf_token() }}">
            <button name="action"
value="accept" class="btn
btn-download">Accept</button>
            <button name="action"
value="reject" class="btn
btn-delete">Reject</button>
        </form>
    </div>
</div>
{%
endfor %}
</div>
{%
else %}
    <p class="no-requests">No pending
requests.</p>
{%
endif %}

{%
with messages =
get_flashed_messages(with_categories=True) %}

```

```
{% if messages %}  
    <div class="flash-messages">  
        <ul>  
            {% for category, msg in messages %}  
                <li class="{{ category }}">{{ msg }}</li>  
            {% endfor %}  
        </ul>  
    </div>  
{% endif %}  
{% endwith %}  
</section>  
</main>  
</body>  
</html>
```

## templates/login.html:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <title>SyncSphere - Login</title>  
    <meta name="viewport"  
content="width=device-width, initial-scale=1.0">  
    <link rel="preconnect"  
href="https://fonts.googleapis.com">  
    <link rel="stylesheet" href="{{ url_for('static', filename='css/auth.css') }}>  
</head>
```

## SyncSphere

```
<body>
    <div class="login-container">
        <div class="logo">
            
            <h1>SyncSphere</h1>
        </div>
        <h2>Access your files securely</h2>
        <form action="{{ url_for('auth.login') }}"
method="post">
            <input type="hidden" name="csrf_token"
value="{{ csrf_token() }}">
            <input type="text" name="username_or_email"
placeholder="Username or Email" required>
            <input type="password" name="password"
placeholder="Enter your password" required>
            <label>
                <input type="checkbox" name="remember_me"
value="on"> Remember Me
            </label>
            <button type="submit">Log in</button>
        </form>
        <p class="sign-up-text">
            Need an account? <a href="{{
url_for('auth.register') }}>Sign Up</a>
        </p>
        {%- with messages =
get_flashed_messages(with_categories=True) %}
        {%- if messages %}
```

```
<div class="flash-messages">
    <ul>
        { % for category, message in messages
        %
            <li class="{{ category }}>{{ message }}</li>
        %
        { % endfor %
    </ul>
</div>
{ % endif %
{ % endwith %
</div>
</body>
</html>
```

**templates/register.html:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>SyncSphere - Register</title>
    <meta name="viewport"
content="width=device-width, initial-scale=1.0">
    <link rel="preconnect"
href="https://fonts.googleapis.com">
    <link rel="stylesheet" href="{{ url_for('static', filename='css/auth.css') }}>
</head>
<body>
```

```
<div class="login-container">
    <div class="logo">
        
            <h1>SyncSphere</h1>
        </div>
        <h2>Create your account</h2>
        <form action="{{ url_for('auth.register') }}" method="post">
            <input type="hidden" name="csrf_token"
value="{{ csrf_token() }}">
            <input type="text" name="username"
placeholder="Choose a username" required>
            <input type="email" name="email"
placeholder="Your email address" required>
            <input type="password" name="password"
placeholder="Create a password" required>
            <button type="submit">Register</button>
        </form>
        <p class="sign-up-text">
            Already have an account? <a href="{{ url_for('auth.login') }}">Log In</a>
        </p>
        {%- with messages =
get_flashed_messages(with_categories=True) %}
        {%- if messages %}
            <div class="flash-messages">
                <ul>
```

```
{% for category, message in messages
%}
    <li class="{{ category }}>{{{
message } }}</li>
    {% endfor %}
</ul>
</div>
{% endif %}
{% endwith %}
</div>
</body>
</html>
```

### static/css/auth.css:

```
:root {
    --primary-color: #4f80e1;
    --text-color: #333;
    --bg-color: #fff;
    --font-family: 'Poppins', sans-serif;
}

* {
    box-sizing: border-box;
}

body {
    margin: 0;
    padding: 0;
```

## SyncSphere

```
font-family: var(--font-family);
background-color: var(--bg-color);
color: var(--text-color);
}

.navbar {
  display: flex;
  align-items: center;
  justify-content: space-between;
  height: 60px;
  padding: 0 2rem;
  background-color: #fff;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.navbar .logo {
  display: flex;
  align-items: center;
}

.navbar .logo img {
  height: 32px;
  margin-right: 8px;
}

.navbar .logo h1 {
  font-size: 1.5rem;
  margin: 0;
}
```

```
.navbar nav ul {  
    list-style: none;  
    display: flex;  
    gap: 1rem;  
    margin: 0;  
    padding: 0;  
}  
  
.navbar nav ul li a {  
    text-decoration: none;  
    color: var(--text-color);  
    font-weight: 500;  
    transition: color 0.3s ease;  
}  
  
.navbar nav ul li a:hover {  
    color: var(--primary-color);  
}  
  
.container {  
    width: 90%;  
    max-width: 1100px;  
    margin: 0 auto;  
    padding: 2rem 0;  
}  
  
.files-section {  
    text-align: center;  
    margin-bottom: 3rem;  
}
```

```
.files-section h1 {  
  font-size: 2rem;  
  margin-bottom: 1rem;  
}  
  
.file-actions {  
  margin-top: 1rem;  
}  
  
.file-actions button {  
  background-color: var(--primary-color);  
  color: #fff;  
  border: none;  
  padding: 0.75rem 1.5rem;  
  margin: 0.5rem;  
  border-radius: 4px;  
  cursor: pointer;  
  font-size: 1rem;  
  transition: opacity 0.3s ease;  
}  
  
.file-actions button:hover {  
  opacity: 0.9;  
}  
  
.hero-section {  
  text-align: center;  
  padding: 3rem 2rem;  
  background-color: #f8f9fc;
```

```
border-top: 1px solid #e2e2e2;
}

.hero-section h2 {
  font-size: 2rem;
  margin-bottom: 1rem;
}

.hero-section p {
  font-size: 1.1rem;
  line-height: 1.6;
  margin-bottom: 2rem;
  max-width: 600px;
  margin-left: auto;
  margin-right: auto;
}

.hero-section button {
  background-color: var(--primary-color);
  color: #fff;
  border: none;
  padding: 1rem 2rem;
  border-radius: 4px;
  cursor: pointer;
  font-size: 1.1rem;
  transition: opacity 0.3s ease;
}

.hero-section button:hover {
  opacity: 0.9;
```

```
}

.footer {
    text-align: center;
    padding: 1rem;
    font-size: 0.9rem;
    background-color: #fff;
    border-top: 1px solid #e2e2e2;
}

.btn {
    background-color: var(--primary-color);
    color: #fff;
    border: none;
    padding: 0.75rem 1.5rem;
    border-radius: 4px;
    cursor: pointer;
}

.btn:hover {
    opacity: 0.9;
}

.login-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
    width: 100%;
```

```
}

.login-container .logo {
  display: flex;
  align-items: center;
  gap: 1rem;
  margin-bottom: 1rem;
}

.login-container .logo img {
  width: 50px;
}

.login-container .logo h1 {
  font-size: 1.5rem;
  margin: 0;
}

.login-container h2 {
  font-size: 1.2rem;
  margin: 1rem 0;
}

.login-container form {
  display: flex;
  flex-direction: column;
  width: 300px;
  gap: 1rem;
}
```

## SyncSphere

```
.login-container form input {  
    padding: 0.75rem;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
}  
  
.login-container form button {  
    background-color: var(--primary-color);  
    color: #fff;  
    border: none;  
    padding: 0.75rem;  
    border-radius: 4px;  
    cursor: pointer;  
    transition: opacity 0.3s ease;  
}  
  
.login-container form button:hover {  
    opacity: 0.9;  
}  
  
.sign-up-text {  
    margin-top: 1rem;  
    font-size: 0.9rem;  
}  
  
.flash-messages ul {  
    list-style: none;  
    padding: 0;  
    margin: 1rem 0;  
}
```

```
.flash-messages li.success {  
    background-color: #d4edda;  
    border: 1px solid #c3e6cb;  
    color: #155724;  
    padding: 0.75rem;  
    border-radius: 4px;  
    margin-bottom: 0.5rem;  
}  
  
.flash-messages li.error {  
    background-color: #ffe7e7;  
    border: 1px solid #ffa3a3;  
    color: #d8000c;  
    padding: 0.75rem;  
    border-radius: 4px;  
    margin-bottom: 0.5rem;  
}
```

### **static/css/dashboard.css:**

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
}  
  
body {
```

```
font-family: "Segoe UI", Tahoma, Geneva,
Verdana, sans-serif;
background-color: #f6f8fa;
color: #333;
}

.navbar {
    display: flex;
    align-items: center;
    justify-content: space-between;
    background-color: #fff;
    padding: 1rem 2rem;
    border-bottom: 1px solid #ddd;
}

.navbar-left {
    display: flex;
    align-items: center;
}

.logo {
    height: 40px;
    margin-right: 0.5rem;
}

.app-name {
    font-size: 1.25rem;
    font-weight: 600;
    color: #222;
}
```

```
.navbar-right a {  
    margin-left: 1.5rem;  
    text-decoration: none;  
    color: #555;  
    font-weight: 500;  
}  
  
.navbar-right a:hover {  
    color: #000;  
}  
  
main {  
    max-width: 1200px;  
    margin: 2rem auto;  
    padding: 0 2rem;  
}  
  
.files-section {  
    background-color: #fff;  
    padding: 2rem;  
    border-radius: 8px;  
    margin-bottom: 2rem;  
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
}  
  
.files-section h2 {  
    font-size: 1.5rem;  
    margin-bottom: 1rem;  
}
```

```
.files-table {  
    width: 100%;  
    border-collapse: collapse;  
    margin-top: 1rem;  
}  
  
.files-table thead {  
    background-color: #f0f0f0;  
}  
  
.files-table th,  
.files-table td {  
    padding: 0.75rem;  
    text-align: left;  
    border: 1px solid #ddd;  
}  
  
.no-files {  
    margin-top: 1rem;  
}  
  
.btn {  
    display: inline-block;  
    padding: 0.5rem 1rem;  
    margin-right: 0.5rem;  
    text-decoration: none;  
    font-size: 0.875rem;  
    border-radius: 4px;  
    cursor: pointer;
```

```
border: none;
}

.btn-download {
    background-color: #007BFF;
    color: #fff;
}

.btn-download:hover {
    background-color: #0056b3;
}

.btn-delete {
    background-color: #dc3545;
    color: #fff;
}

.btn-delete:hover {
    background-color: #c82333;
}

.upload-cta {
    text-align: center;
    background-color: #fff;
    padding: 2rem;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.upload-cta h2 {
```

```
    font-size: 1.5rem;
    margin-bottom: 1rem;
}

.upload-cta p {
    color: #666;
    margin-bottom: 1.5rem;
}

.upload-form {
    display: inline-block;
    text-align: left;
}

.upload-input {
    margin-bottom: 1rem;
}

.btn-upload {
    background-color: #28a745;
    color: #fff;
}

.btn-upload:hover {
    background-color: #218838;
}

.btn-remove {
    background-color: #dc3545;
    color: #fff;
```

```
}

.btn-remove:hover {
    background-color: #c82333;
}

.flash-messages {
    max-width: 1200px;
    margin: 1rem auto;
    padding: 1rem 2rem;
    background-color: #d4edda;
    border: 1px solid #c3e6cb;
    border-radius: 8px;
    color: #155724;
}

.flash-messages ul {
    list-style: none;
}

.flash-messages li.success {
    background-color: #d4edda;
    border: 1px solid #c3e6cb;
    color: #155724;
    padding: 0.75rem;
    border-radius: 4px;
    margin-bottom: 0.5rem;
}

.flash-messages li.error {
```

```
background-color: #d4edda;
border: 1px solid #c3e6cb;
color: #155724;
padding: 0.75rem;
border-radius: 4px;
margin-bottom: 0.5rem;
}

.action-section .action-form {
  display: flex;
  gap: 0.5rem;
}

.action-section .action-form
input[name="to_username"] {
  flex: 1;
  min-width: 0;
  height: 2.5rem;
  font-size: 1.1rem;
  padding: 0.4rem 0.6rem;
  border-radius: 4px;
  border: 1px solid #ccc;
}

.action-section .action-form button.btn-upload {
  height: 2.6rem;
  padding: 0 1.2rem;
  font-size: 1.1rem;
  border-radius: 4px;
}
```

```
.list-section {  
    background-color: #fff;  
    padding: 2rem;  
    border-radius: 8px;  
    max-width: 1000px;  
    margin: 2rem auto;  
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);  
}  
  
.list-section h2 {  
    font-size: 2rem;  
    margin-bottom: 1.5rem;  
    text-align: center;  
}  
  
.friends-grid {  
    display: grid;  
    grid-template-columns: repeat(auto-fill,  
minmax(240px, 1fr));  
    gap: 1.5rem;  
}  
  
.friend-card {  
    background-color: #fdfdfd;  
    padding: 1.5rem;  
    border-radius: 8px;  
    text-align: center;  
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);  
    display: flex;
```

```
flex-direction: column;
align-items: center;
}

.friend-info .friend-name {
  font-size: 1.25rem;
  font-weight: 600;
  margin: 0;
}

.friend-info .friend-username {
  font-size: 1rem;
  color: #555;
  margin: 0.25rem 0;
}

.friend-info .friend-details {
  font-size: 0.875rem;
  color: #777;
  margin: 0.2rem 0;
}

.friend-actions {
  margin-top: auto;
  display: flex;
  gap: 0.5rem;
}

.no-friends {
  text-align: center;
```

```
font-size: 1.1rem;
color: #555;
}

.list-section .action-form {
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 0.5rem;
  margin-bottom: 2rem;
  max-width: 600px;
  margin-left: auto;
  margin-right: auto;
}

.list-section .action-form
input[name="to_username"] {
  flex: 2;
  min-width: 200px;
  height: 2.5rem;
  font-size: 1.1rem;
  padding: 0.4rem 0.6rem;
  border-radius: 4px;
  border: 1px solid #ccc;
}

.list-section .action-form button.btn-upload {
  height: 2.6rem;
  padding: 0 1.2rem;
  font-size: 1.1rem;
```

```
border-radius: 4px;  
}
```

## **קישור לעמוד ה-github שלuproject:**

<https://github.com/eyalp7/SyncSphere>