

שרת CloudMaster

# תיק פרויקט

מגיש אופיר דרי מורה: עופר גלעדי



# מסמך אפיון

מגיש אופיר דרי מורה: עופר גלעדי

## 1.הקדמה

הפרויקט הזה נעשה במסגרת של לימודי סייבר- רשתות ומערכות הפעלה הנוספות ל5 יח"ל מדעי המחשב. הפרויקט הוא בפרויקט גמר בהיקף של 5 יח"ל.

מסמך זה הוא מסמך האפיון של הפרויקט והוא יספק מידע אודות הפרויקט- מה הוא עושה, מהן מטרותיו ויתרונותיו, הפונקציות העיקריות שלו, קהל היעד של הפרויקט וכדומה.

**1.1 מטרה**

במסגרת מגמת המחשבים בבית הספר, אנו לומדים 5 יחידות לימוד במדעי המחשב ו 5 – יחידות לימוד סייבר. לימוד הסייבר בבית הספר מתבצע במסגרת תוכנית גבהים. תוכנית הלימודים בהגנת סייבר פותחה על ידי מטה הסייבר הצה"לי תוך שיתוף הידע הפדגוגי של משרד החינוך.

בתוכנית לימודים זו, נלמדת הגנת רשתות, הגנת מערכות הפעלה ואתיקה. בסוף הלימודים התלמידים כותבים פרויקט גמר בהיקף של כ 5 - יחידות לימוד. מסמך זה הינו מסמך האפיון של פרויקט הגמר, ומטרתו לספק מידע אודות המוצר המפותח: מה הוא עושה, מטרותיו ויתרונותיו, הפונקציות העיקריות אותן הוא מציע, קהל היעד שלו, אילוצים, דרישות וכיוצא בזאת. מסמך זה נועד לאפשר לקורא להבין במה מדובר - מה זה בכלל הפרויקט הזה? קהל היעד של המסמך הינו בוחני בגרות הפרויקטים בהגנת סייבר.

**1.2 המוצר**

שם המוצר CloudMaster.

CloudMaste היא תוכנת אחסון וניגון קבצים. באמצעות מוצר זה יהיה ניתן ללקוח לשמור קבצים במחשב מרוחק באופן פרטי ולגשת אליהם. ניתן לנגן קבצים שנמצאים על השרת.

מטרת המוצר היא הרחבת שטח האיחסון של הלקוח ללא הוספת חומרה פיזית(HDD, SSD).

יתרונותיו העיקריים הם שמירת קבצי הלקוח בנוחות ומהירות.

המוצר חוסך קניית חומרה חיצונית והתקנת ומאפשר הרחבה של שטח האיחסון בנוחות ומהירות.

**1.3 הגדרות**

**מודל שרת לקוח -** המודל מחלק את המשימות או עומס העבודה בין ספק השירות או המשאבים- השרת ,לבין מבקש השירות - הלקוח .שרת-לקוח היא אחת מתצורות ההתקשרות הנפוצות ברשתות

מחשבים .השרת הוא תוכנה פסיבית, המאזינה לרשת ומחכה לקבל בקשות. הלקוח לעומתו בדרך כלל מהווה את ממשק המשתמש - הוא מופעל על ידי המשתמש ופונה לשרת כאשר הוא זקוק למידע

או שירותים ממנו.

**MULTITHREADING -** הוא היכולת של מערכת הפעלה תכנית או לשרת יותר ממשתמש אחד בכל פעם ולנהל בקשות מרובות בו זמנית ללא צורך במספר עותקים של התוכניות הפועלות במחשב. כל יכולה יחידת העיבוד המרכזית לבצוע מספר רב של תהליכים ביעילות. במקרה שאחד הTHRED -ים נעצר מסיבה כלשהי או תקלה מסוימת זה לא מפריע לריצת שאר התוכנית. ה- THREDים יכולים לחלוק ביניהם מידע.

**1.4 תקציר**

המשך מסמך האפיון יכלול את אופי המוצר. יהיה ניתן להכיר אותו, להבין אותו, את דרישותיו, את הפונקציות אותן הוא מציע; במובן הטכני, וכן לא רק. יהיה ניתן להבין את תכלית המוצר, את הרעיון העומד מאחוריו ואת מה שעתיד הוא להציע למשתמש העתידי.

## 2. תיאור כללי

**2.1 פונקציונליות**

סיכום הפונקציות העיקריות במערכת.

**התחברות או הרשמת הלקוח:**

קבלת שם משתמש וסיסמא מהלקוח והעברתם לשרת. במידה וקיים חשבון ללקוח השרת ייתן לו גישה לאחסונו הפרטי. אם אינו קיים ייפתח עבורו חשבון ושטח אחסון פרטי.

**הורדת קובץ:**

קבלת קובץ דרך socket באמצעות קבלת כל החלקים שלו ומיקומו בתוך תיקייה מוכרזת.

**העלאת קובץ:**

העברת כל החלקים של הקובץ דרך socket.

**ניגון קובץ:**

ניגון הקובץ אצל הלקוח.

**2.2 קהל יעד**

כל אדם יכול לגשת למערכת על פי דרישותיה. ממשק המערכת הינו פשוט במיוחד ומאפשר גישה נוחה אליו, ולכל האפשרויות המוצעות דרכו. לא דרושות מיומנויות מיוחדות בתחום הטכנולוגי בקרב משתמשי הממשק. משתמשי הממשק נדרשים לדעת אנגלית בסיסית כאשר זוהי שפת המוצר הראשית. כמו כן, מידע אודות השימוש הטכני במוצר זמין למשתמש בעת השימוש בו, לשם מקסום ידידותיות המשתמש ללקוח.

**2.3 אילוצים עיקריים**

1. **מספר המשתמשים-** המערכת מורכבת מצד שרת ומצד לקוח.
2. **מודולים-**

צד השרת:

א. תמיכה בתקשורת מול לקוחות ב. תמיכה בריבוי לקוחות

צד הלקוח:

א. תמיכה בממשק משתמש לכל לקוח

ב. תמיכה בתקשורת מול שרת

**2.4 הנחות ותלויות**

המערכת תרוץ בסביבת windows ותורץ מתוך Python.

**2.5 סיקור מצב השוק כיום**

כיום, קיים שירות איחסון וניגון קבצים בשרת מרוחק נרחב.

לדוגמא לשירותי איחסון: DropBox, Google drive.

אפרט לדוגמה על אחד שירותי האיחסון, Google Drive. שירות האיחסון הנ"ל מאפשר איחסון קבצים מכל סוג וניגונם.

## 3. דרישות מפורטות

### 3.1 דרישות פונקציונליות

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **מקרי קצה** | **פלט** | **תהליך** | **הסבר**  **קלט** | | **דרישה** |
|  | הצגה בגרפיקה של התוצאה – אם התחבר או לא התחבר במידה ולא עליו לנסות שוב או להירשם | העברת הנתונים לשרת | שם משתמש וסיסמא | קבלת הנתונים ע"י הקלדתם בתיבת טקסט | **קבלת שם משתמש וסיסמת המשתמש** |
|  | הצגת האופציות לפעולות על הקובץ | קריאה לפונקציות האפשרויות עבור הקובץ הנבחר | שם הקובץ | בחירה מתוך אייקונים של הקבצים אשר בשטח האיחסון של הלקוח | **בחירת קובץ משטח האיחסון** |
|  | הצגה בגרפיקה של השלמת ההורדה | העברת הנתונים לשרת וקבלת הקובץ | הקובץ הנבחר והפעולה | כפתור אופציה יוריד את הקובץ מהשרת ללקוח | **הורדת קובץ** |
|  | הרצת הקובץ | העברת הנתונים לשרת, קבלת הקובץ וניגונו. | הקובץ הנבחר והפעולה | כפתור אופציה שינגן קובץ מהשרת . | **ניגון קובץ** |
| 1. קובץ אינו קיים. 2. קובץ אינו מסוג 3MP או 4MP | הצגה גרפית של התוצאה ועידכון הקבצים המופיעים בשרת | העברת הקובץ לשרת | PATH קובץ | תיבת טקסט שתקבל PATH של קובץ קיים בלקוח ותעלהו לשרת | **העלאת קובץ** |

### 3.2 דרישות של ממשקים חיצוניים

3.2.1 ממשקי משתמש )למשל GUI(

תמיכה במודול הממשק.

3.2.2 ממשקי חומרה

רכיביו הפיסיים של המחשב, בסיסיים בלבד: עכבר, מקלדת, מסך.

3.2.3 ממשקי תוכנה )אם התכנה מחולקת לכמה תוכנות שונות שצריכות לתקשר בניהן(

אין דרישות מיוחדות .

3.2.4 ממשקי תקשורת )למשל תקשורת בין שרת ללקוחות(

תמיכה בתקשורת שרת - לקוח .

תמיכה בריבוי לקוחות.

**3.3 דרישות לא פונקציונליות**

3.1.3 ביצועים דרישות

המערכת תממש ביצועי הפונקציות המפורטות מעלה, במהירות האפשרית וביעילות.

3.3.2 מהימנות דרישות

התעבורה תעבור באופן מסודר ומבוקר .

כל קלט יעבור בדיקה מתאימה.

3.3.3 זמינות דרישות

השרת יהיה זמין להקשבה מתמדת לפתיחת תקשורת של לקוח עמו, גם בעת תקשורתו מול לקוח

אחר.

3.3.4 דרישות אבטחה

אין דרישות מיוחדות.

3.3.5 דרישות תחזוקה

אין דרישות מיוחדות.

36.3. ניידות דרישות

אין דרישות מיוחדות

**3.4 דרישות בסיס נתונים**

אין דרישות מיוחדות

**3.3 דרישות נוספות**

אין דרישות נוספות.



CloudMaster

# מסמך עיצוב

מגיש אופיר דרי מורה: עופר גלעדי

## 1.הקדמה

סקירה מהירה של מסמך האפיון תוך התייחסות למוצר אותו פיתחתי.

**1.1 מטרה**

מסמך זה הינו מסמך העיצוב של פרויקט הגמר, ומטרתו לספק מידע אודות ארכיטקטורת המערכת, רכיביה, המודולים בה היא משתמשת, עיצוב, מבני הנתונים וכיוצא בזאת.

קהל היעד של המסמך הינו בוחני בגרות הפרויקטים בהגנת סייבר.

**1.2 המוצר**

**המוצר המפותח: שרת איחסון Cloud Master**

שרת האיחסון אשר בניתי מאפשר ללקוח לאחסן קבצים במחשב מרוחק, גישה נוחה אליהם וניגונם.

לא זאת ועוד השרת מאפשר חיבור למספר לקוחות במקביל ולכן מהווה שרת הנוח לשימוש למספר משתמשים בו זמנית.

**מטרת המוצר העיקרי הינה הרחבת שטח האיחסון של הלקוח בצורה נוחה וללא צורך בחומרה חיצונית(HDD,SSD).**

**1.3 הגדרות**

**מודל שרת לקוח -** המודל מחלק את המשימות או עומס העבודה בין ספק השירות או המשאבים-

השרת ,לבין מבקש השירות - הלקוח .שרת-לקוח היא אחת מתצורות ההתקשרות הנפוצות ברשתות

מחשבים .השרת הוא תוכנה פסיבית, המאזינה לרשת ומחכה לקבל בקשות. הלקוח לעומתו בדרך

כלל מהווה את ממשק המשתמש - הוא מופעל על ידי המשתמש ופונה לשרת כאשר הוא זקוק למידע

או שירותים ממנו .

**Socket -** ממשק תוכנתי להעברת מידע בין תוכנות שונות. זהו API שמסופק בידי מערכת ההפעלה .

**Pickle -** מודול זה שימושי עבור הפרויקט שלי והוא עוזר העברת מידע בין השרת ללקוח. הוא ממיר אובייקט לבתים וכך מאפשר העברתו דרך socket והמרתו חזרה לאובייקט.

**1.4 תקציר**

מסמך עיצוב זה אינו נותן רק את אופיו ותכונותיו של הממשק אותו פיתחתי, הוא

מסביר ומפרט אודות השימושים השונים, התועלת בממשק זה, נקודות הדמיון

בינו לבין ממשקים אחרים המצליחים קיום, הצעותיו למשתמשים, דרך פעולה

קלה, נוחה וברורה.

## 2.אכריטקטורת המערכת

**2.1 מבט על**

שליחת תוצאה ומידע

שליחת הוראה ומידע



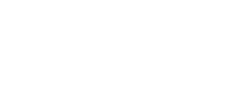
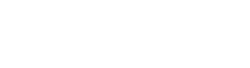
שרת



לקוח



לקוח

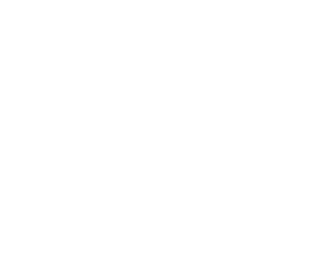


למוצר 2 צדדים – צד השרת וצד הלקוח. הקשר ביניהם:

הלקוח בוחר איזה פעולה הוא רוצה לבצע(הורדה, העלאה, ניגון).

השרת מקבל את בחירת הלקוח ומקבל או מוסר את המידע המבוקש.

צד השרת- השרת יכול לטפל במספר לקוחות במקביל.



לקוח

•

בממשק

תמיכה

.

לקוח

לכל

משתמש

•

בתקשורת

תמיכה

.

שרת

מול

•

תמיכה

תמיכה

רשימה

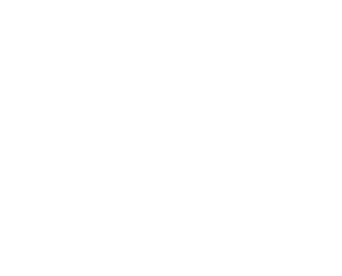
בהעברת

(

בעזרת

pickle

.)



שרת

•

תמיכה

בתקשורת

לקוחות

.

מול

•

בריבוי

תמיכה

.

לקוחות

•

בהעברת

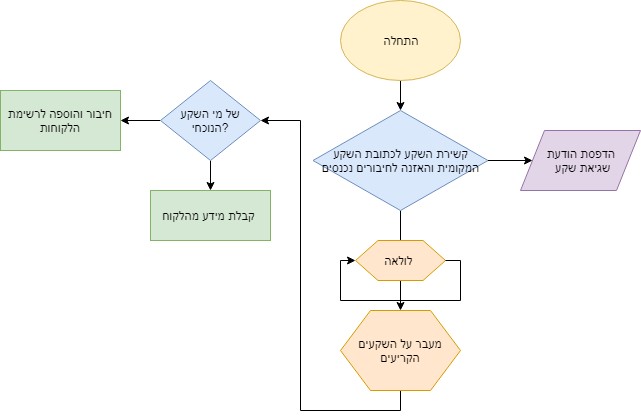
תמיכה

רשימות באמצעות

PICKLE

IMPORT TKINTER :GUI מודל ממשק המשתמש

**צד השרת- השרת יכול לטפל בכמה לקוחות במקביל.**



**מחלקות בשרת:**

**THREADEDSERVER מחלקה**

מרחיבה (OVERLOADING) את מחלקת MULTITHREADING, בונה תהליכון שרץ במקביל שמטרתו לאפשר ללקוחות להתחבר לשרת, ומכיל את הפרמטרים הבאים:

* PLIST – רשימה המכילה את תהליכוני הלקוחות

)LISTENTOCLIENT מסוג(

* HOST – כתובת אבסולוטית של השרת.
* PORT – תת כתובת, פרוטוקול IP..
* Q – תור לשימושו של התהליכון SENDTOCLIENT.
* SOCK – אובייקט מסוג SOCKET .

ואת הפונקציות הבאות:

פונקציית הבנאי-

* מקבלת את הפרמטרים הבאים:
  + HOST – לאיזה כתובות השרת פתוח.
  + PORT – לאיזה פורט השרת מאזין.
  + Q – תור שישתמש בתהליך בן.

מפעילה את הפעולות העיקריות הבאות:

מפעיל את התהליכון SENDTOCLIENT שמאזין בתור להודעות שעליו לשלוח לקליינטים.

קושרת לשרת את כתובת הIP ואת הפורט שעליהם השרת יאזין, קובעת את מספר המשתמשים המקסימלי כאשר לקוח מבקש להתחבר התהליך מקצה לו את התהליכון

.LISTENTOCLIENT מטיפוס

פונקציית RUN-

מופעלת כאשר המתודה START מופעלת על עצם מסוג THREADEDSERVER, לולאה אינסופית שמחכה ללקוחות וכאשר לקוח נכנס היא פותחת עבורו תהליכון LISTENTOCLIENT.

**SENDTOCLIENT מחלקה**

מרחיבה (OVERLOADING) את מחלקת MULTITHREADING, מאפשרת שליחת הודעות מהשרת ללקוחות, ומכילה את התכונות הבאות:

* SIZE – גודל ההודעות שהשרת שולח.
* Q – תור ניהול כלל ההודעות שלא טופלו,

שבין הלקוחות לשרת )כל איבר מכיל רשימה שאיברה הראשון הוא הSOCKET של הלקוח שאליו צריכה להישלח הודעה ואיברה השני הוא תוכן ההודעה שצריכה להישלח(

פונקציית RUN-

מופעלת כאשר המתודה START מופעלת על עצם מסוג SENDTOCLIENT, רצה בלולאה "אינסופית," ומבצעת את הפעולות העיקריות הבאות:

* אם התור Q לא ריק:
  + מוציאה את האיבר בראש התור )רשימה שמכילה נתונים על הודעה שצריכה להישלח לאחד הלקוחות( לתוך משתנה.
  + שולחת את ההודעה ללקוח המתאים.

**LISTENTOCLIENT מחלקה**

מרחיבה (OVERLOADING) את מחלקת MULTITHREADING, מאפשרת את ניהול האפליקציה מול הלקוח ומכילה את התכונות הבאות:

* .הלקוח SOCKET – CLIENT
* ADDRESS – כתובת IP הלקוח.
* PLIST – רשימה המכילה תהליכונים של לקוחות.

)LISTENTOCLIENT מסוג(

* SIZE – גודל ההודעות שהשרת מקבל.
* Q – תור למשלוח הודעות ללקוח.

ואת הפונקציות הבאות:

פונקציית הבנאי-

* מקבלת את הפרמטרים הבאים:
* CLIENT – שם של לקוח.
* ADDRESS – כתובת של לקוח.
* PLIST - רשימה המכילה את תהליכוני הלקוחות )לצורך משחקים בזוגות.(
* Q

פונקציית RUN- מפעילה את SERVERLOGIC.

**פונקציות השייכות גם לשרת וללקוח (general\_functions):**

**rec\_vid:**

**טענת כניסה-** מקבלת עצם סוקט המחובר ללקוח או שרת ומקום אחסון, מקבלת את שם הקובץ וכל החלקים שלו דרך הסוקט.

**טענת יציאה-**  שומרת את הקובץ במקום האחסון ומחזירה TRUE בסיום.

**Upload\_vid:**

**טענת כניסה-** מקבלת עצם סוקט המחובר ללקוח או שרת ושם קובץ.

**טענת יציאה-**  שולחת את הקובץ דרך הסוקט, אינה מחזירה נתונים.

**פונקציות שרת:**

**login**

**טענת כניסה-** מקבלת עצם סוקט המחובר ללקוח.

**טענת יציאה-** מחזירה את נתוני המשתמש(שם,סיסמא,אחסון).

**נועדה לאפשר גישה לנתוני הלקוח.**

**signup**

**טענת כניסה-** מקבלת עצם סוקט המחובר ללקוח ופרטי המשתמש.

**טענת יציאה-** יוצרת מקום אחסון פרטי ללקוח ושומרת את חשבונו במאגר הנתונים ומחזירה את נתוני המשתמש(שם,סיסמא,אחסון).

**נועדה ליצור חשבון משתמש חדש.**

**Send\_files**

**טענת כניסה-** מקבלת עצם סוקט המחובר ללקוח ומקום האחסון של המשתמש.

**טענת יציאה-** שולחת את שמות הקבצים השייכים ללקוח, אינה מחזירה נתונים.

**download**

**טענת כניסה-** מקבלת עצם סוקט המחובר ללקוח ומקום האחסון של המשתמש ושם של קובץ אותו הלקוח מבקש.

**טענת יציאה-** שולחת את הקובץ ללקוח באמצעות קריאה לg\_func.upload\_vid, אינה מחזירה נתונים.

**צד לקוח:**

**פונקציות הלקוח-**

**login**

**טענת כניסה-** מקבלת עצם סוקט, שם משתמש, סיסמא וחלון טקינטר.

**טענת יציאה-** שולחת את הנתונים לשרת ומחזירה את התוצאה- הצליח להתחבר או לא הצליח ובמידה והצליחה היא תסגור את החלון הנוכחי.

**Startup**

**טענת כניסה-** מקבלת את הפעולה שהמשתמש בחר בלחיצה על כפתור.

**טענת יציאה-** קוראת לפונקצית login או signup בהתאם. אם הצליח היא תמשיך לחלון הבא אחרת תציג את השגיאה.

**Signup**

**טענת כניסה-** מקבלת עצם סוקט, שם משתמש, סיסמא וחלון טקינטר.

**טענת יציאה-** שולחת את הנתונים לשרת ומחזירה את התוצאה- הצליח לייצור חשבון או לא הצליח ובמידה והצליחה היא תסגור את החלון הנוכחי.

**Download\_act**

**טענת כניסה-** לא מקבלת דבר.

**טענת יציאה-** משמיעה צליל, לוקחת את שם הקובץ הנבחר וקוראת לפונקציה download.

**Download**

**טענת כניסה-** מקבלת עצם סוקט, מקום אחסון ושם קובץ.

**טענת יציאה-** מורידה את הקובץ למקום האחסון באמצעות קריאה לrec\_vid.

**Click**

**טענת כניסה-** אין.

**טענת יציאה-** משמיע צליל.

**Play**

**טענת כניסה-** לא מקבלת דבר.

**טענת יציאה-** משמיעה צליל, לוקחת את שם הקובץ הנבחר וקוראת לפונקציה play\_a\_file.

**Play\_a\_file**

**טענת כניסה-** מקבלת עצם סוקט, מקום אחסון ושם קובץ.

**טענת יציאה-** במידה והקובץ קיים במחשב הלקוח היא תפעיל את הקובץ, אחרת היאמורידה את הקובץ למקום האחסון באמצעות קריאה לrec\_vid ואז תפעיל אותו.

**Browse\_func**

**טענת כניסה-** לא מקבלת.

**טענת יציאה-** משמיעה צליל, מנקה את תיבת הטקסט ופותחת חלון explorer המאפשר לבחור קובץ מהמחשב.

**Create\_error**

**טענת כניסה-** מקבלת שם שגיאה.

**טענת יציאה-** עוצרת את התוכנית ומראה מסך בו כתוב שם השגיאה.

**Upload**

**טענת כניסה-** לא מקבלת.

**טענת יציאה-** בודקת אם הקובץ בתיבת הטקסט מאוחסן בשרת ובמידה ולא היא תעלה אותו לשרת ותוסיף אותו לתיבת הקבצים.

**Get\_files**

**טענת כניסה-** מקבלת עצם סוקט.

**טענת יציאה-** מחזירה את רשימת שמות הקבצים המאוחסנים בשרת.

**2.3 דיון בנושא העיצוב שנבחר**

הפרויקט נכתב בשפת תכנות עילית - פייתון. פייתון היא שפת תכנות עילית הנמצאת בשימוש נפוץ. מצאתי לנכון לכתוב את הפרויקט בפייתון. ראשית, הפרויקט שלי מבוסס על תקשורת שרת - לקוח, אותה ניתן לממש בפייתון בדרך קצרה וברורה.

נוסף על כך, בניית הממשק העיצובי בשימוש במודול הTKINTER בפייתון היה ברור, מובן ומסודר.

החסרונות בתוכנה הם שיש צורך להתקין הרחבות רבות, ולא פעם נתקלתי בקשיים ובהגבלות באשר לסביבת העבודה.

החלטתי בכל זאת לממש הפרויקט שלי בשפה זו כיוון שיתרונותיה נעלים על חסרונותיה, ואני שמח על החלטה זו שלי.

# 3.עיצוב נתונים ופרוטוקולים

תקשורת בין השרת ללקוח:

**“commands”**

**“ready”**



client



server



## 4. ממשק משתמש

ממשק המשתמש נכתב בTKINTER ומוצג למשתמש באופן נוח, ברור וקל לשימוש!