**ספר פרוייקט:**

**P.E.P -**

**Photo Edit Platform**



בית ספר: הדסים

שם העבודה: P.E.P - Photo Edit Platform

שם התלמיד: עומר עוז

ת.ז. התלמיד: 213419518

שם המנחה: ניר דוויק

תאריך ההגשה: 27.5.2021



**תוכן עניינים –**

1. מבוא............................................................................................5
   1. תיאור תכולת הספר....................................................................5
   2. הרקע לפרוייקט..........................................................................5
   3. תהליך מחקר הפרוייקט...............................................................5
   4. סקירת המצב הקיים בשוק............................................................6
   5. חידושים שיש בפרוייקט...............................................................6
   6. בעיות בכתיבת הפרוייקט ופתרונות.............................................6-7
   7. מוטיבציה לעבודה.....................................................................8
   8. הצורך שעליו הפרוייקט עונה.........................................................8
2. מבנה הפרוייקט.....................................................................................9..
   1. מבנה כללי של הפרוייקט............................................................9..
   2. Front-End.............................................................................10
      1. Login Window........................................................11-10
      2. Edit Window...........................................................16-12
   3. Back-End.............................................................................16
      1. Client.....................................................................18-16
      2. Server.........................................................................18
      3. Edit Image...................................................................18
3. מדריך למשתמש...................................................................................19
   1. הוראות הכנה וקבצים נדרשים.....................................................19
   2. הפעלת הפרוייקט.....................................................................19
   3. תפקיד המסכים........................................................................19
      1. תפקיד מסך ה Login Window.........................................23
      2. תפקיד מסך ה Edit Window............................................23
         1. תפקיד תת המסכים של Edit Window............20-22
   4. הסבר הודעות למשתמש............................................................23
4. בסיס נתונים..........................................................................................24
   1. אופן הפעולה והשמירה בבסיס הנתונים........................................24
   2. שליפת תמונות מבסיס הנתונים...................................................25
5. מדריך למפתח......................................................................................26
   1. Client...................................................................................26
      1. המחלקה ComData.......................................................26
      2. המחלקה ID...................................................................27
      3. Valid\_request..............................................................27
      4. Send\_request\_to\_server.......................................28-29
      5. Receive\_server\_response..........................................30
      6. Handle\_server\_response......................................31-33
   2. המחלקה Login\_window.........................................................34
      1. העלאת תמונה................................................................34
      2. הכנסת ID......................................................................35
   3. המחלקה edit\_window............................................................36
      1. Archive\_image.......................................................36-38
         1. View\_image..................................................39
      2. Request\_edit.........................................................40-42
      3. Update\_image............................................................43
      4. Edit buttons.................................................................44
         1. Win\_crop.......................................................44
         2. Win\_resize....................................................45
         3. Win\_rotate....................................................46
   4. Server..................................................................................47
      1. המחלקה Image….........................................................47
      2. המחלקה RecvData.......................................................48
      3. המחלקה Busy..............................................................49
      4. המחלקה IDlist..............................................................49
      5. Handler\_queue...........................................................50
         1. Check\_client\_request...................................50
         2. Handle\_client\_request..................................50
         3. Send\_response\_to\_client.............................51
      6. Handler\_thread...........................................................51
         1. Receive\_client\_request...........................52-53
      7. Handle\_image\_edit.....................................................54
      8. Send\_image\_to\_client................................................55
      9. Check\_valid\_id......................................................56-58
   5. Edit\_image...........................................................................59
      1. המחלקה ImageTK...................................................59-60
      2. Read\_image...............................................................60
      3. Save\_to\_dir.................................................................61
      4. Editing....................................................................62-65
6. סיכום אישי..........................................................................................66
7. ביבליוגרפיה..........................................................................................67

**1.מבוא**

**1.1 תיאור תכולת הספר -**

ספר זה מתאר את הדרך שבה פיתחתי את -

**P.E.P – Photo Edit Platform**

כיצד התוכנה עובדת, הסיבה שבחרתי לפתח את התוכנה, תהליך המחקר, האתגרים במהלך כתיבת הפרוייקט, מדריך למשתמש ולמפתח והסברים נוספים על הפרוייקט.

**1.2 הרקע לפרויקט -**

יש לי תשוקה לצילום ועריכת תמונות. הנושא הזה תמד עניין אותי ורציתי לחקור אותו לעומק. מתחילת השנה ידעתי שהפרוייקט שלי יהיה קשור לזה, אף על פי שהנושא היה מאוד מעורפל והמליצו לי לקחת את הפרוייקט לכיוון אחר. המוטיבציה שלי בכתיבת הפרוייקט תמיד הייתה גבוהה מכיוון שהנושא מאוד עניין אותי ובער בי לראות שאני מצליח לבנות תוכנת עריכה עם פיצ'רים שאין בתוכנות העריכה המתקדמות ביותר בשוק כיום.

**1.3 תהליך מחקר הפרוייקט -**

בכדי לגשת לבניית הפרוייקט ולהבין מה אני רוצה שיעשה הייתי צריך קודם כל לעשות מחקר ראשוני על מהי תמונה, כיצד היא בוניה וכו'. לדוגמא, ההבדל בין תמורה בפורמט JPG לפורמט PNG הוא שתמונות אשר נשמרות בפורמט JPG לרוב מאבדות קצת מהרזולוציה שלהן בכדי להקטין את גודל התמונה. לעומת זאת, תמונות שנשמרות בפורמט PNG, גודל התמונה לא משתנה כלל. לאחר המחקר הראשוני בדקתי כיצד ניתן לעבוד עם תמונות בפייתון. מצאתי מספר ספריות כגון: pil, pillow, open cv  
לבסוף בחרתי לעבוד עם הספרייה pillow מכיוון שהיא התאימה בדיוק לצורכי. לאחר קריאה ממושכת והרבה כתיבה הגעתי לרמה שאפשרה לי לבצע את כל מה שאני רוצה מבחינת עריכת תמונות ושמירתן בפרוייקט.

**1.4 סקירת המצב הקיים בשוק -**

כיום בשוק יש אין ספור אפליקציות ותוכנות של עריכת תמונות.   
האפליקציות הפופולריות ביותר הן:

LightRoom, SnapSeed, Photo Editor Pro

לכל התוכנות מאפיינים מאוד דומים, יש בהן מגוון אפשרויות של עריכה ושמירת תמונות, ממשק משתמש ידידותי ומודרני ואפקטים מרשימים שניתן להחל על התמונה.

**1.5 חידושים שיש בפרוייקט -**

בפרוייקט שלי קיימים מספר חידושים אשר לא קיימים באף תוכנה אחרת בשוק כיום.

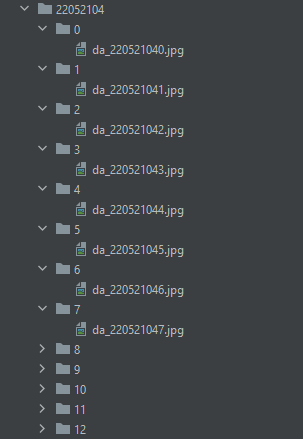
* החידוש הראשון הוא שניתן לערוך תמונה במקביל. מספר לקוחות יכולים להתחבר אחד לשני בו זמנית ולצפות בעריכות של שאר הלקוחות ולשנות אותם.
* החידוש הנוסף הוא שללקוחות (רק בעלי קוד גישה) יש גישה מלאה למאגר התמונות אשר נערכו בתוכנה ונשמרו בממסד הנתונים.

**1.6 בעיות בכתיבת הפרוייקט ופתרונות -**

בפרוייקט שלי אירעו מספר בעיות, בחרתי להתמקד במרכזיות שבהן:

* סינכרוניזציה בין מספר לקוחות אשר רוצים לערוך תמונה –   
  כאשר מספר לקוחות מתחברים בו זמנית ומבקשים לערות תמונה מהשרת, עלולה להיווצר בעיה סינכורנית שהשרת לא ידע איזה עריכה לבצע קודם ואיזה תמונה להציג. לשם כך יצרתי מנגנון אשר יבטיח סינכרוניזציה.   
  כאשר משתמש לוחץ על הכפתור “request edit”, נשלחת לשרת פקודה לבקשת עריכה. גם שליחת הפקודה הזו נעשית בצורה סינכורנית באמצעות תור מסונכרן אשר פועל ללא הפסקה ב thread נפרד.  
  השרת יחזיר תגובה חיובית ויאפשר ללקוח לערוך את התמונה רק בתנאי שאין לקוח אחר שלחץ על הכפתור “request edit” ועורך אותה. כך רק לקוח אחד יוכל לבקש בקשת עריכה מהסרבר ולמנוע בעיות סנכרון.

* הצגת תמונת ארכיון מבסיס הנתונים - בכדי להציג ללקוח תמונה מבסיס הנתונים הייתי צריך לשמור את התמונות בצורה שיהיה אפשר לגשת אליהן לאחר מכן. ראשית, יצרתי לכל לקוח אשר יוצר חיבור עם השרת ID ייחודי משלו, וכל גרסה של תמונה תקבל מספר. בבסיס הנתונים יצרתי לכל ID חדש תיקייה, ותיקיות נוספות בתוכה לכל גרסה של התמונה.   
  כך כל התמונות נשמרו באופן מסודר ויכולתי לקרוא להן בקלות. דוגמא לתיקייה בבסיס הנתונים:



תמונה 1

**1.7 מוטיבציה לעבודה -**

בחרתי נושא לפרוייקט שמאוד מעניין אותי, ולכן תמיד הייתה לי מוטיבציה ונהנתי לכתוב את הפרוייקט, למרות כל האתגרים והקשיים שאירעו במהלך הכתיבה למדתי המון על הנושא.  
בנוסף, בסיטואציות שבהן הייתי חסר אונים ולא יודע איך להתקדם, תמיד המורה שלי תמך ביוישב איתי עד שאפתור את הבעיה, לא משנה כמה זמן הדבר יקח.

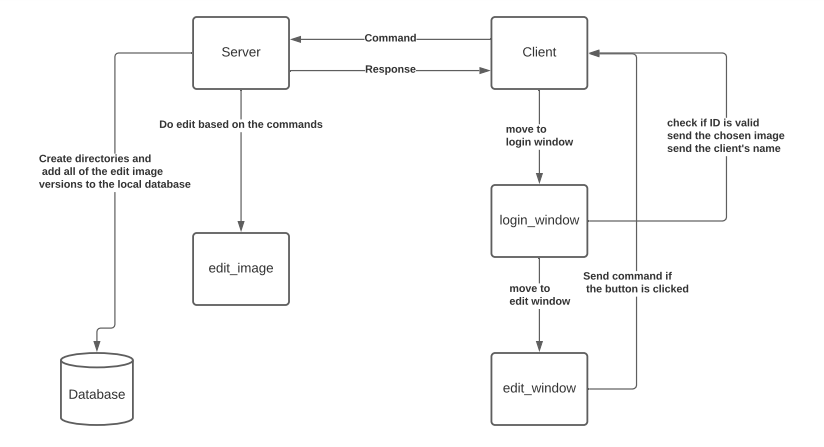
**1.8 הצורך שעליו הפרוייקט עונה -**

עריכת התמונות היא חלק בלתי נפרד מהחיים שלנו. התוכנה יכולה להתאים במיוחד לקבוצות\חברות אשר צריכות לערוך ביחד תמונה ספציפית. שכן הדבר יקל עליהם ויחסוך זמן יקר.

1. **מבנה הפרוייקט**

הפרוייקט הוא תוכנה שיתופית לעריכת תמונות. מספר לקוחות יכולים לערוך אותה תמונה ולצפות בשינויים שנעשו על ידי כל לקוח. כל העריכות מתבצעות בצד השרת כך שמספר לקוחות יכולים לערוך את אותה התמונה.

הפרוייקט מחולק לשני חלקים: חלק ה Front-End (קדמי) וחלק ה Back-End (אחורי). חלק הFront-End אחראי על האינטרקציה של המשתמש עם התוכנה. חלק ה Back-End. אחראי על העברת המידע והפקודות, עריכת התמונות, שליחתן ושמירתן.

**2.0 מבנה כללי של הפרוייקט -**

תמונה 2

**2.1 Front-End**

חלק זה מכיל מספר קבצים שמטרתם להציג את התוכנה למשתמש בדרך ידידותית. המשתמש מעלה תמונה, בוחר את העריכות ורואה אותם על המסך.

**2.1.1 Login Window**

הפרוייקט נפתח בחלון המשתמש. (ראה תמונה 3) בחלון זה המשתמש מכניס את שמו, התמונה שאותה הוא רוצה לערוך ולוחץ על SUBMIT. כאשר הכפתור נלחץ, הלקוח מעביר לשרת את התמונה ומחכה לקבל ממנו אישור שהתמונה התקבלה. לאחר מכן הלקוח יכנס למצב חלון העריכה.

לחילופין המשתמש יכול להכניס ID ובמידה והID קיים כעת במערכת אצל השרת, הלקוח יעבור לחלון העריכה עם התמונה הזהה לחלון העריכה אצל הלקוח הראשון.   
כאשר הID הוכנס, השרת בודק האם ID זה קיים במערכת כעת. במידה וכן, השרת יחזיר ללקוח פקודה שהID ואלידי ויציג ללקוח את התמונה של הלקוח האחר בעל אותו הID.  
במידה והID לא ואלידי, השרת יחזיר ללקוח תשובה שלילית ויופיע במסך בקשה להכניס קוד ID חדש (ראה תמונה 4)



תמונה 3



תמונה 4

**2.1.2 Edit Window**

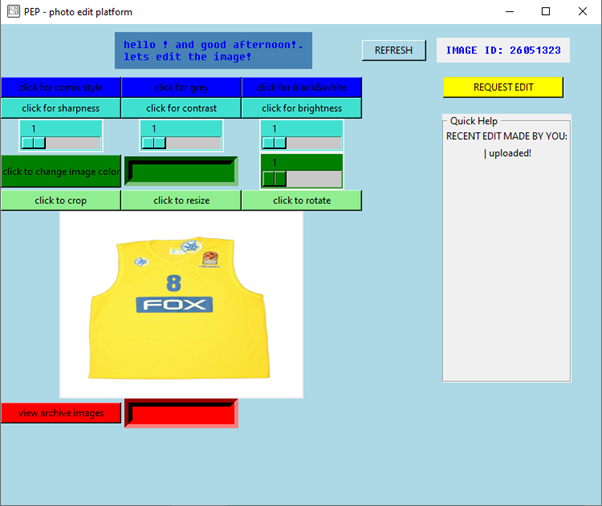
בחלון העריכה מוצג כפתור ריפרש, בקשת עריכה, ID של התמונה, כפתורי העריכה, התמונה עצמה וכפתור לצפות בתמונות בארכיון (ראה תמונה 5).

כאשר הלקוח רוצה לערוך תמונה, ראשית הוא צריך ללחוץ על כפתור בקשת העריכה. במידה והלקוח יכול לערוך, הכפתורים עוברים למצב שבו ניתן להקליק עליהם.

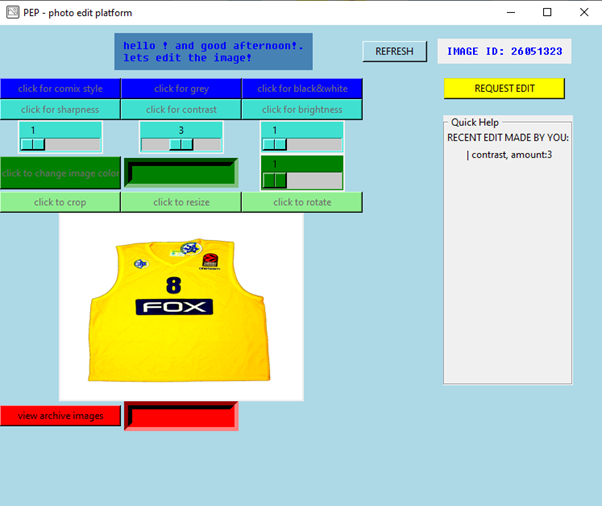
כפתורי העריכה הזמינים למשתמש הם:

* Comix
* Grey
* Black&White
* Sharpness (3 levels)
* Contrast (3 levels)
* Brightness (3 levels)
* Change image color (3 levels)
  + Red
  + Green
  + Blue
  + Purple
  + Yellow
  + Turquoise
* Crop
* Resize
* Rotate

קיימים חלונות נוספים בחלון העריכה בשביל להקל על המשתמש ולהפוך את חווית העריכה לידידותית יותר.



תמונה 5



תמונה 6

3.ביצוע עריכה

כאשר המשתמש לוחץ על כפתור לעריכה. בעת הלחיצה, הלקוח שולח פקודה לעריכה לשרת. אל הפקודה הנשלחת מתווסף פרוטוקול שיעזור לשרת לקרוא את הפקודה במלואה.

השרת מקבל את פרוטוקול הפקודה, מפרק אותו, מבצע את העריכה על התמונה, מעדכן את התמונה אצלו במערכת, שומר אותה ואת הגרסה שלה בממסד הנתונים ומחזיר את התמונה העדכנית (ראה תמונה 5).

הלקוח מציג את התמונה.

**PROTOCOL FORMAT –**

The protocol builder function is located in the client side under the name – “send request to server”.

the protocol format looks like the following –

COMMAND  #

COMMAND LENGTH #

EDIT EXTRA 1 (OPTIONAL) / IMG SIZE (JUST IF THE COMMAND IS “SENDIMG”   #

EDIT EXTRA 2 (OPTIONAL)  / IMG NAME (JUST IF THE COMMAND IS “SENDIMG” #

FOR EXAMPLE –

RECVIMG#07#

GREY#03#

SENDIMG#07#95723#dog.jpg#

SHARPNESS#09#4#

COLOR#05#RED#2#

CROP#04#200#250#

ROTATE#06#30#

**2.2 Back-End**

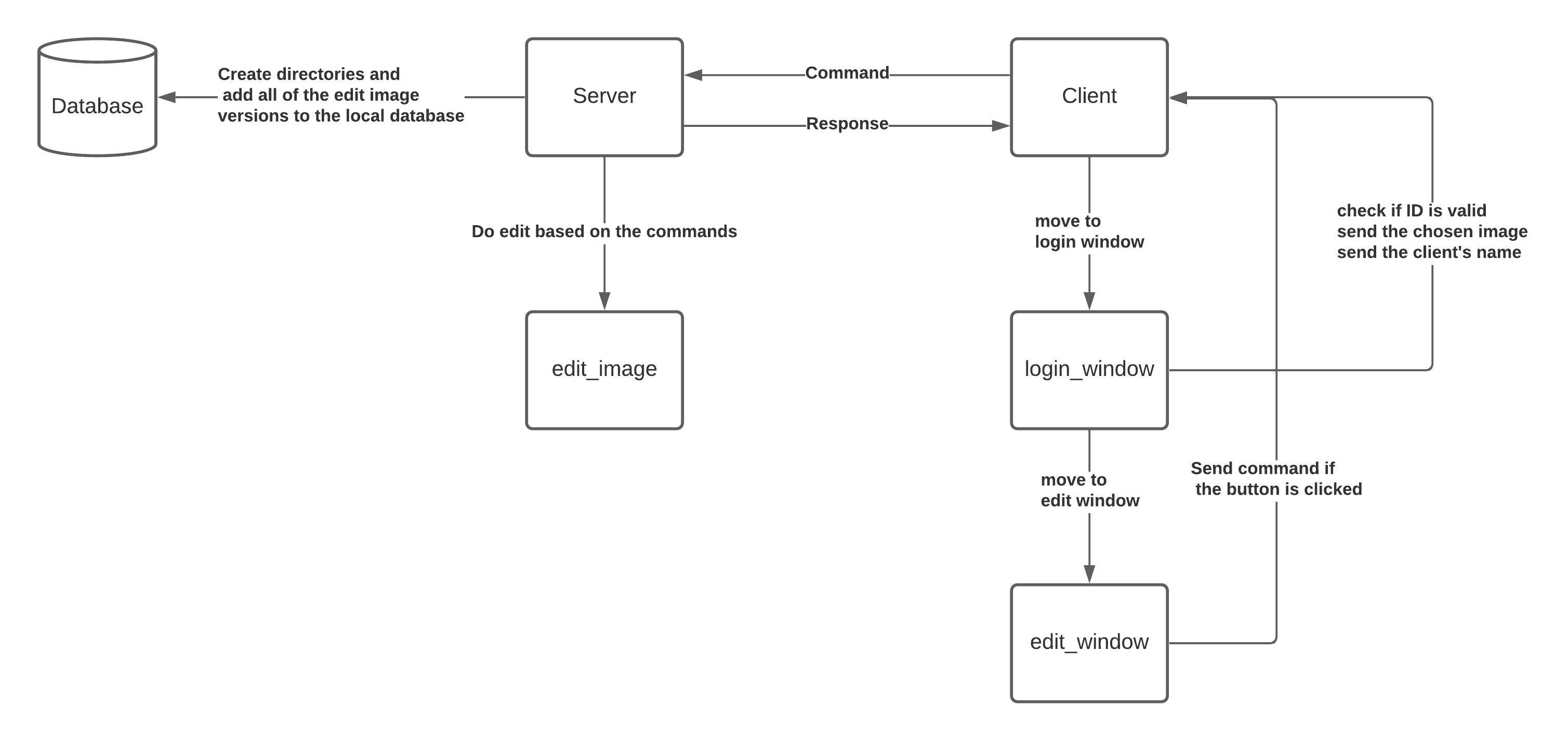
חלק זה מכיל את הלקוח, השרת וקובץ העריכה edit\_image. הלקוח לא נמצא באינטרקציה עם קבצים אלו.

**2.2.1 Client**

הלקוח אחראי על הצגת ממשק המשתמש, ושליחת הפקודות לשרת. הפקודות אשר הלקוח שולח לשרת בזמן הפעולה הן:

* "exit"
* "sendimg"
* "admin"
* "recvimg"
* "reqedit"
* "comix"
* "grey"
* "bw"
* "sharpness"
* "contrast"
* "brightness"
* "color"
* "crop"
* "resize"
* "rotate"
* "vid"
* "id"
* "view"

התקשורת בין הלקוח לממשקי המשתמש נעשים בצורה הזו:  
לאחר הבקשות של הלקוח אל הסרבר, השרת מחזיר response שבאמצעותו הלקוח יודע שהפקודה בוצעה בהצלחה.



AVAILABLE RESPONSES:

* "exit"
* "sent"
* "omer"
* "recv"
* "xrcv"
* "novw"
* “okid”
* “vald”
* “unvd”
* "view"

**2.2.2 Server**

צד הסרבר אחראי לבצע את הפעולות הנשלחות אליו על ידי הלקוחות. הסרבר מאזין לכל לקוח, מקבל את הפקודה שלו, מבצע אותה ומחזיר תגובה.   
בין הפעולות שהשרת מבצע:

* בדיקה אם ה ID שהוכנס על ידי הלקוח הוא ולידי
* בדיקה האם ללקוח יש גישה לעריכה כעת
* שמירת התמונות וגרסאותיהן בבסיס הנתונים
* עריכת התמונות ושליחתם בחזרה ללקוח (נעשה על ידי הקובץ (edit image

**2.2.3 edit\_image**

קובץ זה אחראי על כל העריכות האפשריות של התמונה, כולל קריאת תמונה ושמירתה.

בקובץ זה קיימות פעולות רבות אשר מקבלות תמונה ומחזירות את התמונה הערוכה.

**3. מדריך למשתמש**

**3.1 הוראות התקנה וקבצים נדרשים**

בשביל להריץ את הפרוייקט נדרש תוכנת פייתון בגרסת 3 ומעלה

ואת הקובץ gallery.ico אשר הכרחי לממשק המשתמש.



**3.2 הפעלת הפרוייקט**

בכדי להפעיל את הפרוייקט, בצד הלקוח צריך להיות הקבצים:

* Client
* Login\_window
* Edit\_window

ובצד הסרבר צריך להיות הקבצים:

* Server
* Edit\_image

**3.3 תפקיד המסכים**

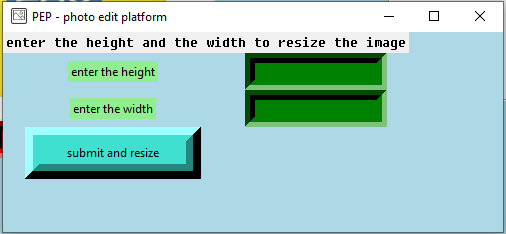
**3.3.1 תפקיד מסך ה Login Window**

תפקיד מסך ה login window הוא לאפשר למשתמש להעלות תמונה \ להכניס ID בשביל לצפות בתמונה קיימת ולערוך אותה.

**3.3.2 תפקיד מסך ה Edit Window**

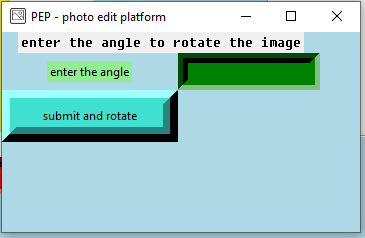
תפקיד מסך זה הוא לאפשר למשתמש לערוך את התמונות, לצפות ב ID של התמונה ולצפות בתמונות מארכיון

**3.3.2.1 תפקיד תת המסכים של - Edit Window**

**Resize Window** 

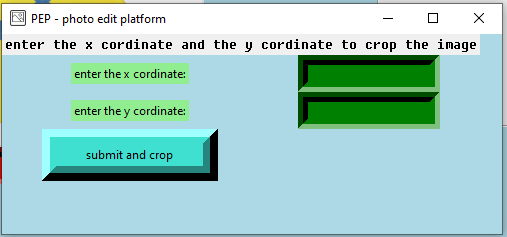
תמונה 7

חלון זה מאפשר לשנות את גודל התמונה

 **Rotate Window**

תמונה 8

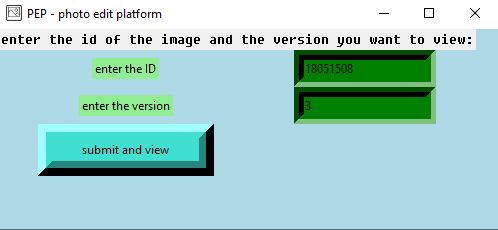
חלון זה מאפשר להטות את התמונה.

**Crop Window**

תמונה 9

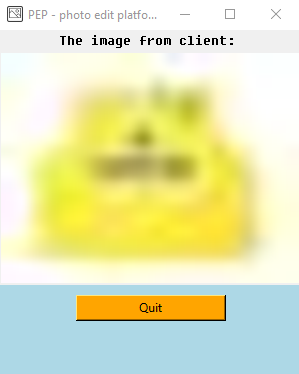
חלון זה מאפשר לחתוך את התמונה

**Archive window**



תמונה 10

חלון זה מאפשר לצפות בתמונה מארכיון התמונות של בסיס הנתונים

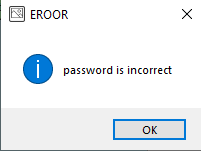


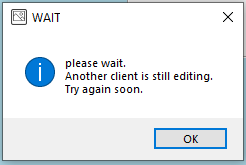
תמונה 11 - התמונה הערוכה המתקבלת לאחר בקשת התמונה מהארכיון

**3.4 הסבר הודעות למשתמש**

קיימות מספר הודעות למשתמש במהלך הפעלת התוכנה:  
(ראה תמונה 14,15,16).



תמונה 12- במידה והלקוח הכניס ID אשר לא נמצא במערכת של השרת, יודפס על המסך הודעת שגיאה בתחתית המסך



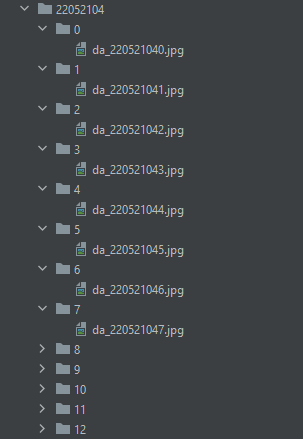
תמונה 13 - במידה והסיסמא הדרושה לצפייה בתמונות מהארכיון לא נכונה, יופיע החלון הקופץ הנל

תמונה 14- במידה ולקוח לוחץ על כפתור בקשת העריכה בזמן שלקוח אחר עדיין עורך את התמונה, יופיע החלון הקופץ הנל

**4. בסיס נתונים**

**4.1 אופן הפעולה והשמירה בבסיס הנתונים**

שמירת התמונה בבסיס הנתונים נעשית בצד של השרת.  
לכל לקוח אשר יוצר חיבור עם השרת ID ייחודי משלו, וכל גרסה של תמונה תקבל מספר. בבסיס הנתונים קיים לכל ID חדש תיקייה, ותיקיות נוספות בתוכה לכל גרסה של התמונה.   
כך כל התמונות נשמרו באופן מסודר (ראה תמונה 15).

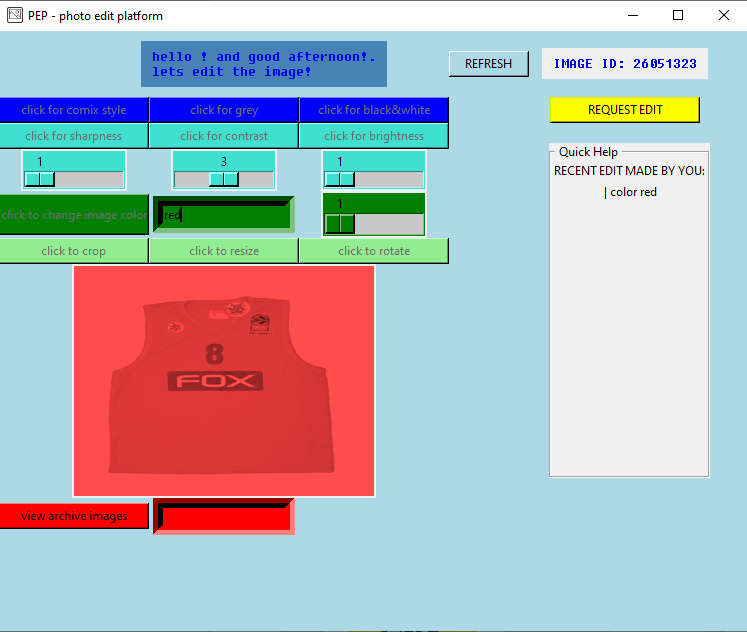


תמונה 15 - דוגמא תיקייה בבסיס הנתונים

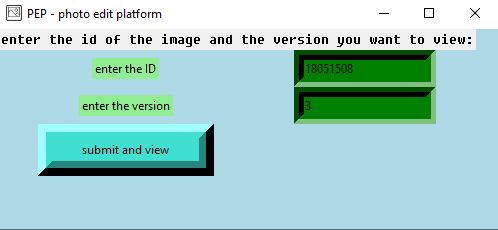
**4.2 שליפת תמונות מבסיס הנתוניםarchive window))**

שליפת תמונות מבסיס הנתונים נעשות באמצעות הכפתור:  
view archive images.

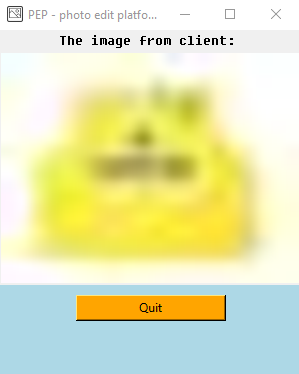
לצד הכפתור יש אפשרות להכנסת סיסמא (ראה תמונה 18 ). רק במידה והלקוח מכניס סיסמא נכונה יפתח חלון נוסף ובו המשתמש יצטרך להכניס את ה ID של התמונה ומספר הגרסה (ראה תמונה 19). לאחר מכן המשתמש יוכל לצפות בתמונה מבסיס הנתונים (ראה תמונה 20).



תמונה 16



תמונה 17



תמונה 18

**5. מדריך למפתח**

**5.1 Client**

הקובץ Client אחראי על התקשורת עם השרת.   
המחלקה שולחת לשרת פקודות ומידע ומקבלת ממנו responses.   
גם המחלקות Login\_window ו edit\_window שולחות בקשות לסרבר באמצעות הקליינט.

ישנם 2 מחלקות בקובץ Client. הראשונה ComData והשנייה ID. בנוסף להם ישנם פעולות רבות .

המחלקה ComData היא מחלקה סינכרונית אשר מרכזת בתוכה את המידע שישלח לשרת. המחלקה פועלת עם הפעולה send\_request\_to\_server ב

המחלקה ID אחראית על שליחת ה ID של התמונה לשרת

**5.1.1 המחלקה ComData**

המחלקה ComData (קיצור של command data), תפקידה לרכז את המידע אשר ישלח לשרת. המחלקה יוצרת אובייקט אשר מקבל 3 פרמטרים:

* Command
* Image data
* Edit extras

כל האובייקטים של המחלקה נכנסים לתור ששמו “qcomdata” בכדי ליצור סינכרוניזציה.   
כאשר התור לא ריק, הלקוח יקרא את המידע ששמור באובייקט של המחלקה ComData, יפרק אותו וישלח אותו כפרוטוקול לשרת באמצעות הפעולה **send\_request\_to\_server.**

**5.1.2 המחלקה ID**

המחלקה ID אחראית על שמירת ה ID של התמונה והגרסה. באמצעותה הסרבר ידע לתייג את התמונה ולהעבירה לבסיס הנתונים. בנוסף, הID והגרסה יצורפו לשם התמונה אשר תישמר אצל הלקוח.

**5.1.3 valid\_request**

הפעולה בודקת אם ה command שהוכנס כאובייקט של ComData ואלידי באמצעות רשימה של פקודות אשר הכנסתי מראש. במידה והפקודה לא ברשימת הפקודות תוחזר שגיאה.

**5.1.4 send\_request\_to\_server**

הפעולה לוקחת את הפקודה, התמונה (במידה וקיימת) ותוספי העריכה ויוצרת פרוטוקול שישלח לשרת.

הפרוטוקול נבנה בצורה הבאה:

COMMAND  #

COMMAND LENGTH #

EDIT EXTRA 1 (OPTIONAL) / IMG SIZE (JUST IF THE COMMAND IS “SENDIMG”   #

EDIT EXTRA 2 (OPTIONAL)  / IMG NAME (JUST IF THE COMMAND IS “SENDIMG” #

FOR EXAMPLE –

RECVIMG#07#

GREY#03#

SENDIMG#07#95723#dog.jpg#

SHARPNESS#09#4#

COLOR#05#RED#2#

CROP#04#200#250#

ROTATE#06#30#

**הפונקציה send\_request\_to\_server**

def send\_request\_to\_server(self):  
 *"""  
 send the request to the server  
 protocol builder  
 """* # in recvimg theres an option to send the verson of the image to get the specific img  
 cmd\_length = len(self.request)  
 if self.request == "sendimg":  
 img\_size = self.get\_num\_pixels(self.filename)  
 logging.info("img length is {}".format(img\_size))  
 data = self.request + "#" + "0" + str(cmd\_length) + "#" + str(img\_size) + "#" + \  
 self.filename.split('\\')[-1] + "#"  
 logging.info("the protocol format of the request: {}".format(data))  
 self.client\_socket.send(data.encode())  
 self.client\_socket.send(self.img\_data)  
 else:  
 if self.edit\_extras is not None: # build a protocol with the extra variables for the edit  
 # edit extras -> extra var for the edit (color, level, amount).  
 if len(self.edit\_extras) == 2:  
 if cmd\_length < 10:  
 data = self.request + "#" + "0" + str(cmd\_length) + "#" + str(self.edit\_extras[0]) + \  
 "#" + str(self.edit\_extras[1]) + "#"  
 else:  
 data = self.request + "#" + str(cmd\_length) + "#" + str(self.edit\_extras[0]) + "#" + \  
 str(self.edit\_extras[1]) + "#"  
 elif len(self.edit\_extras) == 1:  
 if cmd\_length < 10:  
 data = self.request + "#" + "0" + str(cmd\_length) + "#" + str(self.edit\_extras[0]) + "#"  
 else:  
 data = self.request + "#" + str(cmd\_length) + "#" + str(self.edit\_extras[0]) + "#"  
 else:  
 if cmd\_length < 10:  
 data = self.request + "#" + "0" + str(cmd\_length) + "#"  
 else:  
 data = self.request + "#" + str(cmd\_length) + "#"  
 logging.info("the protocol format of the request: {}".format(data))  
 self.client\_socket.send(data.encode())

**5.1.5 receive\_server\_response**

הפעולה מקבלת מילה בעלת 4 אותיות מהשרת

התגובות האפשריות לקל מהשרת הן:

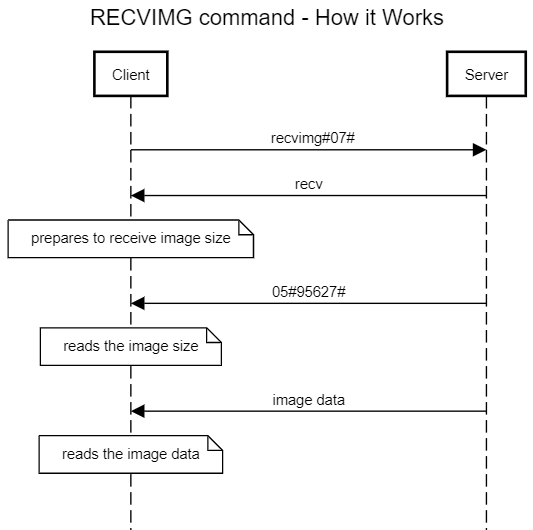
* "exit"
* "sent"
* "omer"
* "recv"
* "xrcv"
* "novw"
* “okid”
* “vald”
* “unvd”
* "view"

**5.1.6 handle\_server\_response**

פעולה זו מבצעת פעולה כלשהי בהתאם לresponse.

לדוגמא כאשר הresponse is “recv”, הלקוח מבקש מהשרת מידע נוסף לגבי התמונה, יוצר קובץ של מונה חדשה ושולח את המידע של התמונה אשר התקבל מהשרת אל תוך הקובץ.

התרשים של התהליך נראה כך:



תמונה 19

**הפונקציה handle\_server\_response**

def handle\_server\_response(self, client\_socket, response):  
 *"""  
 deals with the protocol.  
 this func takes from the request the image size and length size  
 xrcv -> the "recv" command especially for the archived images.  
 """* logging.info("the server's response: {}".format(response))  
 if response == "recv":  
 len\_in\_size = client\_socket.recv(2)  
 len\_in\_size = len\_in\_size.decode()  
 client\_socket.recv(1) # get rid of the first #  
 logging.info("len is:{}".format(len\_in\_size))  
 img\_size = client\_socket.recv(int(len\_in\_size))  
 img\_size = img\_size.decode()  
 client\_socket.recv(1) # get rid of the last #  
 logging.info("img\_size is:{}".format(img\_size))  
 self.client\_image\_name = 'c3\_img\_to\_client\_ver\_{}.jpg'.format(self.image\_count)  
 logging.info("client image name: {}".format(self.client\_image\_name))  
 with open(self.client\_image\_name, 'ab') as f:  
 data = client\_socket.recv(int(img\_size))  
 # count += len(data) # (for debugging purposes)  
 f.write(data)  
 # self.image\_from\_server = open(self.client\_image\_name, 'rb')  
 logging.info('image received to the client!')  
 self.image\_received()  
 self.image\_count += 1  
 if self.request == "reqedit":  
 self.response = self.receive\_server\_response(self.client\_socket)  
 if self.response == "strt":  
 self.start\_edit = True  
 elif self.response == "wait":  
 self.wait\_for\_edit = True  
  
 elif response == "xrcv": # xrcv -> the "recv" command especially for the archived images.  
 self.archive\_id = client\_socket.recv(8) # archive image id  
 self.archive\_id = self.archive\_id.decode()  
 client\_socket.recv(1) # get rid of the first #  
 self.archive\_version = client\_socket.recv(2) # archive image id  
 self.archive\_version = self.archive\_version.decode()  
 client\_socket.recv(1) # get rid of the first #  
 len\_in\_size = client\_socket.recv(2)  
 len\_in\_size = len\_in\_size.decode()  
 client\_socket.recv(1) # get rid of the first #  
 img\_size = client\_socket.recv(int(len\_in\_size))  
 img\_size = img\_size.decode()  
 client\_socket.recv(1) # get rid of the last #  
 archive\_img\_name = 'archived\_{}\_{}.jpg'.format(self.archive\_id, self.archive\_version)  
 with open(archive\_img\_name, 'ab') as f:  
 data = client\_socket.recv(int(img\_size))  
 f.write(data)  
 self.receive\_view\_request = True  
 self.archived\_image = open(archive\_img\_name, 'rb')  
 logging.info('archived image received to the client!')  
  
 elif response == "sent":  
 self.send\_image\_approval = True  
  
 elif response == "strt":  
 self.start\_edit = True  
  
 elif response == "wait":  
 self.wait\_for\_edit = True  
  
 elif response == "vald": # if the id is vald, prepare to receive the image from the other client.  
 len\_in\_size = client\_socket.recv(2)  
 len\_in\_size = len\_in\_size.decode()  
 client\_socket.recv(1) # get rid of the first #  
 logging.info("len is:{}".format(len\_in\_size))  
 img\_size = client\_socket.recv(int(len\_in\_size))  
 img\_size = img\_size.decode()  
 client\_socket.recv(1) # get rid of the last #  
 logging.info("img\_size is:{}".format(img\_size))  
 self.client\_image\_name = 'c3\_img\_to\_client\_ver\_{}.jpg'.format(self.image\_count)  
 with open(self.client\_image\_name, 'ab') as f:  
 data = client\_socket.recv(int(img\_size))  
 # count += len(data) # (for debugging purposes)  
 f.write(data)  
 self.image\_count += 1  
 # self.image\_from\_server = open(self.client\_image\_name, 'rb')  
 logging.info('image received to the client!')  
 self.valid\_id = True  
 self.image\_received()  
  
 elif response == "unvd":  
 self.wrong\_id = True  
  
 elif response == "novw":  
 self.no\_archive\_image = True  
  
 elif response == "exit":  
 client\_socket.close()

**5.2 המחלקה login\_window**

הפרוייקט נפתח בחלון המשתמש. יש 2 אפשרויות:

* העלאת תמונה
* הכנסת ID

**5.2.1 העלאת תמונה**

באפשרות א' המשתמש מכניס את שמו, התמונה שאותה הוא רוצה לערוך ולוחץ על אישור. כאשר הכפתור נלחץ, הלקוח מעביר לשרת את התמונה ומחכה לקבל ממנו אישור שהתמונה התקבלה. לאחר מכן הלקוח יכנס למצב חלון העריכה.

כאשר הלקוח לוחץ על כפתור ה SUBMIT, הלקוח שולח לשרת פקודה sendimg (send image). לאחר פקודה זו, הלקוח שולח את המידע של התמונה הנבחרה לשרת.

השרת מקבל מהלקוח את המידע ושומר את התמונה במערכת של השרת תחת השם: image\_to\_server.

**הפונקציה file\_dialog**

def file\_dialog(self):  
 *"""  
 creates the unique id  
 open the image file the client wanted  
 sets the image path, name, type  
 """* date = datetime.datetime.now()  
 new\_img.set\_id(date.strftime('%d') + date.strftime('%m') + date.strftime('%H') + date.strftime('%M'))  
 file\_name = filedialog.askopenfilename(initialdir="/", title="Select A File",  
 filetype=(("jpeg", "\*.jpg"), ("png", "\*.png")))  
 upload\_lbl = Label(text="the image {} uploaded successfully!\nclick the 'lets go' button to start edit!" \  
 .format(file\_name.split('/')[-1]))  
 full\_img\_name = file\_name.split('/')[-1]  
 upload\_lbl.grid(row=5, column=0, pady=15)  
 correct\_path = file\_name.replace("/", "\\")  
 new\_img.set\_path(correct\_path) # C:\da.jpg  
 # latest path: C:\Users\omero\PycharmProjects\pythonProject6\28041041\0  
 new\_img.set\_name(full\_img\_name.split('.')[0])  
 new\_img.set\_type(full\_img\_name.split('.')[-1])  
 self.client.send\_image(filename=new\_img.get\_path())

**5.2.2 הכנסת ID**

באפשרות ב' המשתמש מכניס ID ובמידה וה ID קיים כעת במערכת אצל השרת, הלקוח יעבור לחלון העריכה עם התמונה הזהה לחלון העריכה אצל הלקוח הראשון.   
כאשר הID הוכנס, הלקוח שולח פרוטוקול עם בקשה view ומצורף לה ה ID שהלקוח הכניס. השרת מקבל את הבקשה ובודק האם הID קיים במערכת באמצעות הפעולה check\_valid\_id() (ראה 5.4.9). במידה והID נמצא במערכת, השרת ישלח ללקוח את התמונה העדכנית של אותו ID והמשתמש יעבור למסך העריכה. במידה והID לא ואלידי, השרת יחזיר ללקוח תשובה שלילית ויופיע במסך בקשה להכניס קוד ID חדש.

**הפונקציה submit\_unknown\_id**

def submit\_unknown\_id(self):  
 id\_list = [self.image\_id.get()]  
 self.client.set\_request\_data("id", None, id\_list)  
 wU = True  
 while wU:  
 if self.client.valid\_id: # checks the condition  
 # NEED TO SEND THE RECENT IMAGE TO MOVE\_TO\_WIN1  
 self.get\_image()  
  
 move\_to\_win1(self.root, self.name, self.client, image=self.image\_server)  
 wU = False  
 self.client.valid\_id = False  
 if self.client.wrong\_id: # checks the condition  
 # SHOW A MESSAGE TO THE CLIENT TO ENTER ID AGAIN  
 error\_lbl = Label(text="ID not found. please try to enter another ID\nOr upload your own image.")  
 error\_lbl.grid(row=6, column=0, pady=15)  
 wU = False  
 self.client.wrong\_id = False

**הפונקציה submit\_valid\_id**

def submit\_valid\_id(self):  
 id\_list = [int(new\_img.get\_id())]  
 self.client.set\_request\_data("vid", None, id\_list)  
 move\_to\_win1(self.root, self.name, self.client, None)

**5.3 המחלקה edit\_window**

מחלקה זו אחראית על כל העריכה של התמונה וצפייה בתמונות מהארכיון.

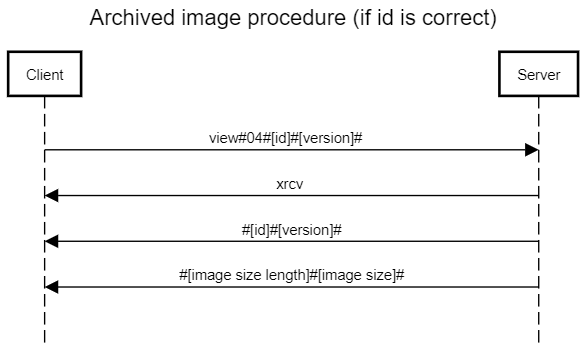
**5.3.1 archive\_image**

הפעולה archive\_image מטרתה לשלוף תמונה מבסיס הנתונים ולהציג אותה ללקוח. הגישה לבסיס הנתונים באמצעות סיסמא בלבד.

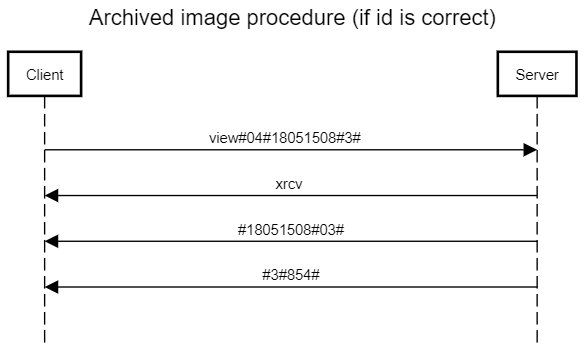
כאשר לקוח מכניס ID וגרסה, נשלח פרוטוקול של בקשת הצפייה, ID וגרסה (בשביל לקבל תמונה ספציפית) לשרת.

השרת בודק אם קיימת תמונה תחת ה ID והגרסה של הלקוח. במידה וכן קיימת, יפתח חלון חדש עם התמונה. במידה ולא קיימת, תוצג הודעה קופצת של שגיאה.

תרשים במידה וה ID משתייך לתמונה (תרשים כללי ותרשים עם ID מדוייק):

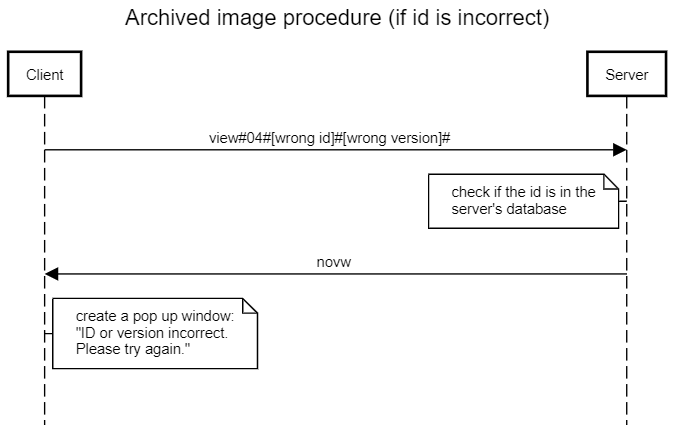
****

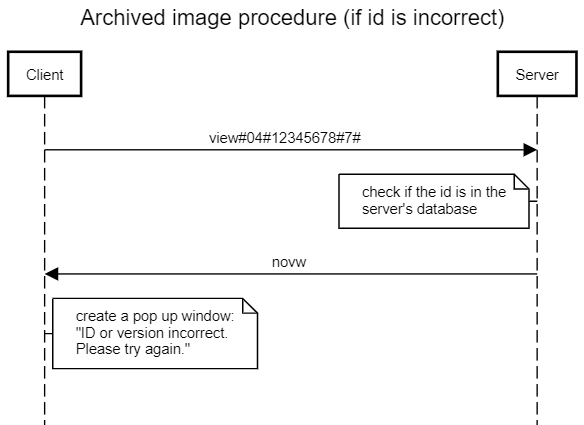
תמונה 20

****

תמונה 21

תרשים במידה וה ID לא משתייך לתמונה (תרשים כללי ותרשים עם ID מדוייק):

****

****תמונה 22

תמונה 23

**5.3.1.1 view\_image**

פעולה זו אחראית על פתיחת חלון חדש שבו תוצג התמונה מבסיס הנתונים. בחלון זה תופיע התמונה וכפתור יציאה לחזרה למסך העריכה.

**הפונקציה view\_image**

def view\_image(self, password):  
 *"""  
 creates a popup window to rotate the image  
 """* if password == "ILovePython": # the password for viewing archived posts.  
 global win\_view  
 win\_view = Toplevel()  
 win\_view.title("PEP - photo edit platform")  
 win\_view.iconbitmap('gallery.ico')  
 win\_view.geometry("500x200")  
 win\_view["bg"] = "light blue"  
 view\_lbl = Label(win\_view, text="enter the id of the image and the version you want to view:")  
 view\_lbl.config(font=("Fixedsys", 10))  
 view\_lbl.grid(row=0, column=0, columnspan=2)  
 view\_lbl\_1 = Label(win\_view, text="enter the ID", bg="light green")  
 view\_lbl\_1.grid(row=1, column=0)  
 id\_entry = Entry(win\_view, fg="black", bg="green", borderwidth=10, width=20)  
 id\_entry.grid(row=1, column=1)  
 view\_lbl\_2 = Label(win\_view, text="enter the version", bg="light green")  
 view\_lbl\_2.grid(row=2, column=0)  
 version\_entry = Entry(win\_view, fg="black", bg="green", borderwidth=10, width=20)  
 version\_entry.grid(row=2, column=1)  
 view\_button = Button(win\_view, text="submit and view",  
 command=lambda: self.do\_view(int(id\_entry.get()), int(version\_entry.get())),  
 fg="black", bg="turquoise", width=20, borderwidth=15)  
 view\_button.grid(row=3, column=0)  
 else:  
 showinfo("EROOR", "password is incorrect")

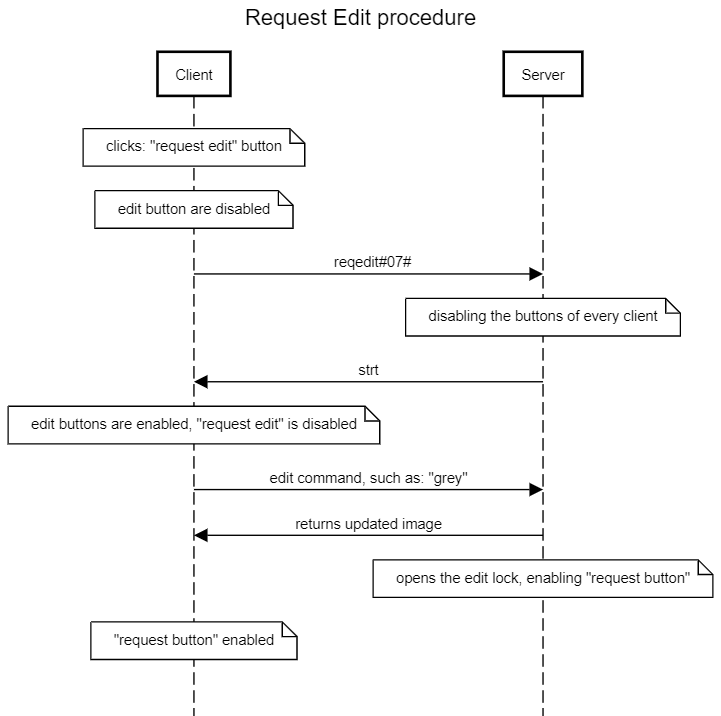
**5.3.2 request\_edit**

בשביל למנוע התנגשויות כל לקוח צריך ללחוץ על הכפתור “request edit” על מנת לערוך את התמונה. כאשר כפתור זה נלחץ, נשלחת בקשה לשרת לבדוק האם קיים לקוח אשר עורך את התמונה ברגע זה.

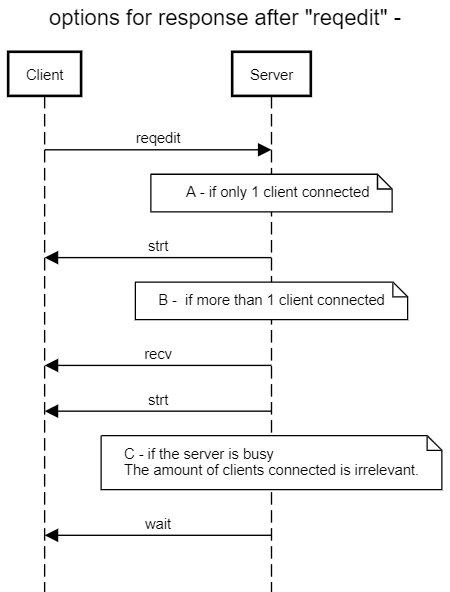
במידה ואין לקוח כזה, כפתורי העריכה עוברים למצב שניתן להקליק עליהם והלקוח יכול לערוך את התמונה. כאשר הלקוח סיים לערוך, הכפתורים חוזרים למצב שלא ניתן להקליק עליהם.

במידה ולקוח עורך את התמונה, תוצג חלונית קופצת שלקוח עורך את התמונה ולנסות שוב מאוחר יותר.

להלן תרשים של אופן פעולת הבקשה:

****

תמונה 24

****

תמונה 25

**הפעולה do\_edit**

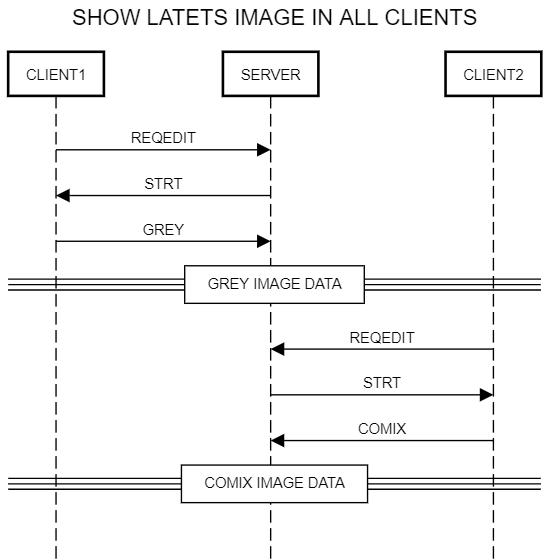
def do\_edit(self):  
 *"""  
 handles the request edit button.  
 self.client.start\_edit -> when True, the recent image is shown in the gui, enable all edit buttons.  
 self.client.wait\_for\_edit -> when True, a popup window of wiat is shown, disable all edit buttons.* ***:return****:  
 """* self.client.set\_request\_data("reqedit", None, None)  
 wU = True  
 while wU == True:  
 if self.client.start\_edit == True: # checks the condition  
 self.image\_server = open(self.client.get\_client\_image\_name(), 'rb')  
 self.img = edit\_image.read\_image(self.image\_server)  
 self.x\_divided = self.new\_img.get\_xsize() / 300  
 self.shown\_image\_ysize = int(new\_img.get\_ysize() / self.x\_divided)  
 self.shown\_image = ImageTk.PhotoImage(  
 edit\_image.resize\_image\_start(self.img, 300, self.shown\_image\_ysize))  
 self.img\_lbl = Label(self.win1, image=self.shown\_image)  
 self.img\_lbl.grid(row=8, columnspan=3)  
 self.enable\_all\_buttons()  
 logging.info("server accepted for this client to edit")  
 wU = False  
 if self.client.wait\_for\_edit == True: # checks the condition  
 self.disable\_all\_buttons()  
 # self.b\_request\_edit["state"] = "disabled"  
 showinfo("WAIT", "please wait.\nAnother client is still editing.\nTry again soon.")  
 logging.info("someone else is editing now")  
 self.client.wait\_for\_edit = False  
 wU = False

**5.3.3 update\_image**

פעולה זו אחראית לעדכן את התמונה העדכנית ביותר במסך העריכה של המשתמש. פעולה זו תעדכן את התמונה במסך לאחר עריכה לוקלית (המשתמש עצמו עורך את התמונה) וגם לאחר שמשתמש אחר עורך את התמונה.

כאשר משתמש אחר עורך את התמונה, בעת לחיצה על הכפתור “request edit”, השרת ישלח לכל לקוח את המידע של התמונה העדכנית ביותר, הלקוחות יקראו את התמונה ובעזרת הפעולה update\_image, התמונה תוצג במסך.

להלן תרשים המציג את אופן הפעולה של לקוח אשר מבצע עריכה:

****

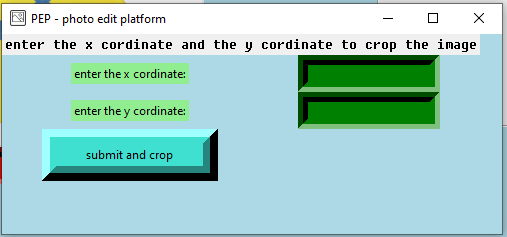
תמונה 26

**5.3.4 edit\_buttons**

החלק המרכזי של מסך העריכה הוא כפתורי העריכה אשר משמשים את הלקוח בעריכת התמונה. בעת לחיצה על כל כפתור, הלקוח קורא לפונקציה אשר תשלח את הבקשה לעריכה לסרבר ותעדכן את התמונה (בעזרת הפעולה update image) ברגע שהתמונה התקבלה.

**5.3.4.1 win\_crop**

חלון אשר נפתח בעת לחיצה על כפתור החיתוך.   
נועד להפוך את העריכה לידידותית יותר.

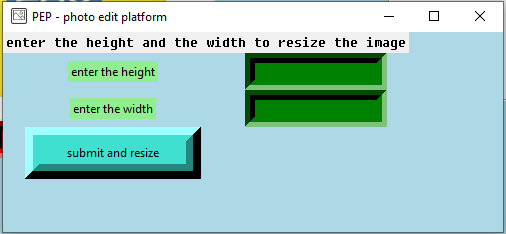
כאשר המשתמש לוחץ על אישור העריכה, נשלח לשרת בקשת העריכה והנתונים הנדרשים בשביל העריכה הספציפית.

תמונה 27

**5.3.4.2 win\_resize**

חלון אשר נפתח בעת לחיצה על כפתור החיתוך.   
נועד להפוך את העריכה לידידותית יותר.

כאשר המשתמש לוחץ על אישור העריכה, נשלח לשרת בקשת העריכה והנתונים הנדרשים בשביל העריכה הספציפית.

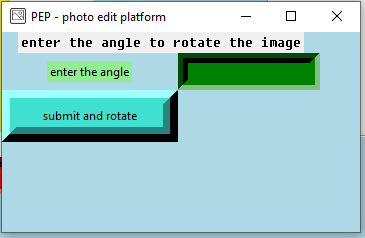


תמונה 28

**5.3.4.3 win\_rotate**

חלון אשר נפתח בעת לחיצה על כפתור החיתוך.   
נועד להפוך את העריכה לידידותית יותר.

כאשר המשתמש לוחץ על אישור העריכה, נשלח לשרת בקשת העריכה והנתונים הנדרשים בשביל העריכה הספציפית.



תמונה 29

**5.4 Server**

השרת אחראי לבצע את כל הבקשות המתקבלות מהלקוחות.

הסרבר מאזין לכל לקוח, מקבל את הפקודה שלו, מבצע אותה ומחזיר תגובה.   
בין הפעולות שהשרת מבצע:

* בדיקה אם ה ID שהוכנס על ידי הלקוח הוא ולידי
* בדיקה האם ללקוח יש גישה לעריכה כעת
* שמירת התמונות וגרסאותיהן בבסיס הנתונים
* עריכת התמונות ושליחתם בחזרה ללקוח (נעשה על ידי הקובץ (edit image

**5.4.1 המחלקה Image**

מחלקה זו אחראית לשמור נתונים על התמונה שהתקבלה ולשלוח אותם למספר פעולות.   
התכונות שנמצאות במחלקה הן:

* Image size
* Image (origina)l
* The recent image
* Image name

במחלקה מוגדרות פעולות get ו set בכדי לעדכן את התכונות ולשלוח אותם.

**5.4.2 המחלקה RecvData**

מחלקה זו אחראית לקבל את כל המידע המקבל מפרוטוקול הלקוח.

הפרמטרים שהמחלקה מקבלת:

* Command
* Command length
* Image data
* Image name
* First extra (for editing proposes)
* Second extra (for editing proposes)
* Client socket

בפעולה handler\_thread() קיימת לולאה אין סופית אשר מאזינה ללקוח. במידה והלקוח שלח פרוטוקול הפרודה receive\_cleint\_request() תפרק את הפרוטוקול ותכניס את הפרמטרים אשר התקבלו לתוך אובייקט המחלקה.

בנוסף, כל האובייקטים של המחלקה נשמרים בתוך תור למטרות סינכרוניזציה. שם התור: DATA\_Q.

**5.4.3 המחלקה Busy**

מחלקה זו אחראית לבדוק אם לקוח עורך כעת.

למחלקה רק תכונה אחת is\_busy

במידה ולקוח לוחץ על כפתור request edit, המשתנה עובר למצב True וכך מונע מלקוחות אחרים לערוך את התמונה.

**5.4.4 המחלקה IDlist**

מחלקה זו אחראית על יצירת רשימה של כל ה ID האקטיבים כעת ורשימת הלקוחות אשר מחוברים לאותו ID.

למחלקה מספר תכונות:

* Id list
* Id count
* Client list
* Client count

כאשר לקוח חדש מתחבר ויוצר תקשורת עם הסרבר, ה ID של אותו לקוח נכנס לרשימה וה client socket שלו גם כן נכנס לרשימת ה client socket list.

כאשר לקוח חדש מכניס ID בחלון הכניסה, הסרבר בודק אם ה ID קיים ברשימת ה ID של המחלקה.

במידה וישנם כמה לקוחות שמחוברים לאותה תמונה. עבור כל עריכה השרת ישלח את התמונה העדכנית לכל הלקוחות ברשימת ה client socket list.

**5.4.5 handler queue**

פעולה זו רצה בthread נפרד על מנת שהיא תעבוד ללא הפסקה.

בפעולה זו יש לולאה אינסופית שמחכה לאובייקט שיכנס לתור data\_q. במידה ואובייקט מסוג RecvData נכנס אל ה data\_q, הפעולה תפרק את כל הפרמטרים של האובייקט ותכניס אותם למשתנים.

לאחר מכן הפעולה תבדוק אם הפקודה תקינה, תבצע את הפעולה הקשורה לפקודה ותחזיר response לקליינט.

בתוך הפעולה קיימת קריאה לשלוש פעולות:

check\_client\_request, handle\_client\_request, send\_response\_to\_client

הפעולות האלו אחראיות לטפל באובייקטים של המחלקה RecvData בצורה סינכרונית ולא תלותית בשאר הפעולות (לאור העובדה ש handler\_queue רצצ על thread נפרד).

**5.4.5.1 check\_client\_request**

הפעולה בודקת אם ה command שהוכנס כאובייקט של RecvData תקין באמצעות רשימה של פקודות אשר הכנסתי מראש. במידה והפקודה לא ברשימת הפקודות תוחזר שגיאה.

**5.4.5.2 handle\_client\_request**

הפעולה מטפלת בבקשות “sendimg” “reqedit” ומחזירה מילה בעלת 4 אותיות. הטיפול בשאר הבקשות יעשה בפעולה הבאה: send\_response\_to\_client

**5.4.5.3 send\_response\_to\_client**

הפעולה מטפלת בפקודות “recv”, “view”, “vid”, “id” וכמו כן בכל פקודות העריכה.

כאשר מתקבלת פקודת עריכה, הפונקציה שולחת ללקוח את התגובה “recv”, המכינה את הלקוח לקבלת תמונה.

לאחר מכן הפונקציה קוראת לפעולה handle\_image\_edit אשר תערוך את התמונה ותשלח אותה.

בנוסף, הפעולה שולחת את ה response ללקוח.

**5.4.6 handler\_thread**

זוהי פעולה אשר רצה על thread נפרד בכדי שתפעל ללא הפסקה במקביל לשאר הפעולות שבשרת.

בפעולה זו קיימת לולאה אין סופית אשר קוראת לפעולה

receive\_client\_request. במידה ולקוח שלח פרוטוקול כלשהו, הפעולה receive\_client\_request תפרק את הפרוטוקול ותעביר את המידע למשתנים אשר יוכנסו בתור אובייקט של RecvData. לאחר מכן האובייקט יכנס לתור התור data\_q.

**5.4.6.1 receive\_client\_request**

הפעולה מפרקת את הפרוטוקול של הלקוח (שלפעמים הוא ארוך מאוד). ומעבירה את כל המידע הנחוץ לאובייקט RecvData. הפעולה רצה עד שהיא פוגשת בסולמית, לאחר כל סולמית המידע נשמר ומועבר למשתנה.

במידה וקיימות פקודות מיוחדות שהעבירו מידע ספיציפי כגון “sendimg” אשר העביר תמונה, או “color” אשר העביר צבע, הפעולה יודעת מה מספר המשתנים אשר הוכנסו בפרוטוקול ומוציאה אותם לפי הסולמיות שתוחמות את המידע.

def receive\_client\_request(client\_socket):  
 *"""  
 receives from the client:  
 command -> request from client. should be in the COMMAND list.  
 size -> the size of the image (if the command is 'sendimg') or the size of the command itself.  
 data -> the image data when the command is 'sendimg'.* ***:param*** *client\_socket: the client socket  
 """* global image\_obj  
 command = ""  
 cmd\_size = ""  
 data = ""  
 image\_name = ""  
 img\_size = ""  
 first\_extra = ""  
 second\_extra = ""  
 data = data.encode()  
  
 command\_tav = None  
 while command\_tav != '#':  
 command\_tav = client\_socket.recv(1)  
 command\_tav = command\_tav.decode()  
 command += command\_tav  
 command = command[:-1]  
 size\_command\_tav = None  
 while size\_command\_tav != '#':  
 size\_command\_tav = client\_socket.recv(1)  
 size\_command\_tav = size\_command\_tav.decode()  
 cmd\_size += size\_command\_tav  
 cmd\_size = cmd\_size[:-1]  
 if command == "sendimg": # need to extract image size and image name  
 size\_image\_tav = None  
 while size\_image\_tav != '#':  
 size\_image\_tav = client\_socket.recv(1)  
 size\_image\_tav = size\_image\_tav.decode()  
 img\_size += size\_image\_tav  
 img\_size = img\_size[:-1]  
 image\_obj.set\_image\_size(img\_size)  
  
 img\_name\_tav = None  
 while img\_name\_tav != '#':  
 img\_name\_tav = client\_socket.recv(1)  
 img\_name\_tav = img\_name\_tav.decode()  
 image\_name += img\_name\_tav  
 image\_name = image\_name[:-1]  
 image\_obj.set\_image\_name(image\_name)  
  
 count = 0 # receiving the image data from the client  
 while count < int(img\_size):  
 semi\_data = client\_socket.recv(1024)  
 data += semi\_data  
 count += len(semi\_data)  
  
 return command, int(cmd\_size), data, image\_name, None, None  
  
 elif command == "color" or command == "crop" or command == "resize" or command == "view":  
 # need to extract 2 edit extras  
 first\_extra\_tav = None  
 while first\_extra\_tav != '#':  
 first\_extra\_tav = client\_socket.recv(1)  
 first\_extra\_tav = first\_extra\_tav.decode()  
 first\_extra += first\_extra\_tav  
 first\_extra = first\_extra[:-1]  
  
 second\_extra\_tav = None  
 while second\_extra\_tav != '#':  
 second\_extra\_tav = client\_socket.recv(1)  
 second\_extra\_tav = second\_extra\_tav.decode()  
 second\_extra += second\_extra\_tav  
 second\_extra = second\_extra[:-1]  
  
 return command, int(cmd\_size), None, None, first\_extra, second\_extra  
  
 elif command == "sharpness" or command == "brightness" or command == "contrast" or command == "rotate" \  
 or command == "vid" or command == "id": # need to extract one edit extra  
 first\_extra\_tav = None  
 while first\_extra\_tav != '#':  
 first\_extra\_tav = client\_socket.recv(1)  
 first\_extra\_tav = first\_extra\_tav.decode()  
 first\_extra += first\_extra\_tav  
 first\_extra = first\_extra[:-1]  
  
 return command, int(cmd\_size), None, None, first\_extra, None  
  
 logging.info("THE COMMAND IS: '{}'. THE IMAGE SIZE IS: {}".format(command, img\_size))  
 return command, int(cmd\_size), None, None, None, None

**5.4.7 handle\_image\_edit**

פעולה זו מקבלת את העריכה שהלקוח העביר.

הפעולה מוצאת את העריכה, ועורכת את התמונה בעזרת הקובץ edit\_image.

לאחר מכן הפעולה קוראת לפונקציה send\_image\_to\_client ומעבירה לה את התמונה הערוכה והסוקט של הלקוח כפרמטר.

def handle\_image\_edit(request, client\_socket, extra1, extra2):  
 if request == "comix":  
 comix\_image = edit\_image.comix\_image(image\_obj.get\_image())  
 send\_image\_to\_client(comix\_image, client\_socket)  
 if request == "grey":  
 grey\_image = edit\_image.gray\_image(image\_obj.get\_image())  
 send\_image\_to\_client(grey\_image, client\_socket)  
 if request == "bw":  
 bw\_image = edit\_image.bw\_image(image\_obj.get\_image())  
 send\_image\_to\_client(bw\_image, client\_socket)  
 if request == "contrast":  
 contrast\_image = edit\_image.contrast\_image(image\_obj.get\_image(), amount=extra1)  
 send\_image\_to\_client(contrast\_image, client\_socket)  
 if request == "brightness":  
 brightness\_image = edit\_image.brightness\_image(image\_obj.get\_image(), amount=extra1)  
 send\_image\_to\_client(brightness\_image, client\_socket)  
 if request == "sharpness":  
 sharpness\_image = edit\_image.sharpness\_image(image\_obj.get\_image(), amount=extra1)  
 send\_image\_to\_client(sharpness\_image, client\_socket)  
 if request == "color":  
 color\_image = edit\_image.change\_image\_color(image\_obj.get\_image(), color=extra1, level=extra2)  
 send\_image\_to\_client(color\_image, client\_socket)  
 if request == "crop":  
 crop\_image = edit\_image.crop\_image(image\_obj.get\_image(), 0, 0, xbottom=extra1, ybottom=extra2)  
 send\_image\_to\_client(crop\_image, client\_socket)  
 if request == "resize":  
 resize\_image = edit\_image.resize\_image(image\_obj.get\_image(), height=extra1, width=extra2)  
 send\_image\_to\_client(resize\_image, client\_socket)  
 if request == "rotate":  
 rotate\_image = edit\_image.rotate\_image(image\_obj.get\_image(), angle=extra1)  
 send\_image\_to\_client(rotate\_image, client\_socket)

**5.4.8 send\_image\_to\_client**

פעולה זו הופכת את התמונה שהתקבלה כפרמטר לפורמט של בתים בכדי שיהיה אפשר לשלוח אותם.

לאחר מכן הפעולה שולחת ללקוח את גודל התמונה ואורך הגודל שלה.

הפעולה שולחת את המידע של התמונה ללקוח ומעדכנת את המחלקה Busy שהעריכה הושלמה וכעת לקוחות אחרים יכולים לערוך את התמונה.

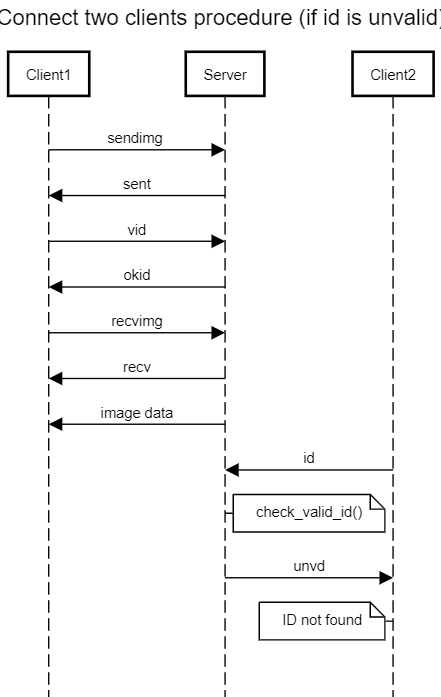
def send\_image\_to\_client(image, client\_socket):  
 *"""  
 send image to the server with command 'recvimg'  
 """* img\_byte\_arr = io.BytesIO()  
 image\_obj.set\_recent\_image(image)  
 image.save(img\_byte\_arr, format='PNG')  
 img\_byte\_arr = img\_byte\_arr.getvalue()  
 length = len(str(int(len(img\_byte\_arr))))  
 size = int(len(img\_byte\_arr))  
 logging.info("the image size is: {}".format(int(len(img\_byte\_arr))))  
 if length < 10:  
 data = "0" + str(length) + "#" + str(size) + "#"  
 else:  
 data = str(length) + "#" + str(size) + "#"  
 client\_socket.send(data.encode())  
 logging.info('sending to the server img size data: {}'.format(data))  
 count = 0  
 while count < int(len(img\_byte\_arr)):  
 if count - int(len(img\_byte\_arr)) < 1024:  
 client\_socket.send(img\_byte\_arr)  
 break  
 client\_socket.send(img\_byte\_arr[0:1024])  
 img\_byte\_arr = img\_byte\_arr[1024:]  
 count += int(len(img\_byte\_arr[0:1024]))  
 b1.stop\_edit()  
 logging.info("image sent successfully to the client!")

**5.4.9 check\_valid\_id**

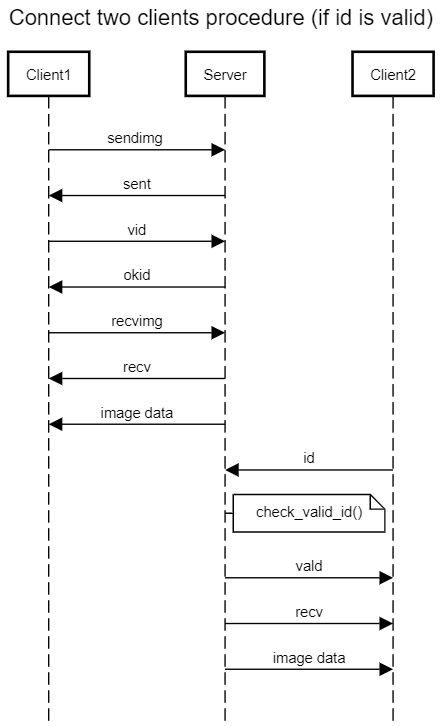
כאשר הלקוח הכניס ID בחלון הכניסה, הוא שולח פרוטוקול של המילה "id" ושל ה ID שהכניס. כאשר השרת מקבל את הפקודה “id”, הוא בודק אם ה id תקין על ידי הפעולה check\_valid\_request.

במידה וה ID נמצא ברשימת ה id\_list, השרת יחזיר אישור, את הפקודה “vald”וישלח ללקוח את התמונה העדכנית ביותר בשרת.

במידה וה ID לא נמצא ברשימת ה id\_list, השרת יחזיר את הפקודה “unvd” שאומרת שה ID לא תקין.



תמונה 30

****

**5.5 edit\_image**

בקובץ זה מתקיים כל העריכות של התמונות, כל העריכות נעשו בעזרת הספרייה pillow.

**5.5.1 המחלקה ImageTK**

המחלקה ImageTK אחראית לשמור את כל המידע המתקבל של התמונה.

התכונות של המחלקה:

* self.path = the image path given
* self.img\_name = the full original image name
* self.full\_name = the full image name in the id directories
* self.latest\_path = the latest edit's path of the image
* self.name =just the name of the image
* self.type = the image type (example: jpg)
* self.admin = the admin image's name
* self.xsize = the width of the image
* self.ysize = the height of the image
* self.id = the unique id of the image (time based)
* self.version\_number = the version of the edit
* self.save\_count = number of times the photo saved to the local computer

**5.5.2 read\_image**

הפעולה קוראת את התמונה ומעבירה אותה לסוג PIL.

**5.5.3 save\_to\_dir**

הפעולה מקבלת תמונה, ID, ו path.

הפעולה שומרת את התמונה בpath ובתוספת ה ID שניתן כפרמטר.

def save\_to\_dir(image, id, image\_name):  
 *"""  
 saves the image in the specific id's directory  
 iname: the image original name (deni)  
 iend: the image type (jpg)  
 id: image's id  
 this\_version: the version of the edit  
 main\_id\_path: example -'C://Users//omero//PycharmProjects//pythonProject6//28012147'  
 full\_path: example - 'C://Users//omero//PycharmProjects//pythonProject6//28012147//1'  
 full name: example - C://Users//omero//PycharmProjects//pythonProject6//28012209//0/da\_280122090.jpg* ***:return****:  
 """* path = os.getcwd()  
 if image\_name is not None:  
 iname = image\_name.split('.')[0]  
 iend = image\_name.split('.')[1]  
 new\_img.set\_name(iname)  
 new\_img.set\_type(iend)  
 if id is not None:  
 new\_img.set\_id(id)  
 this\_version = new\_img.get\_version\_number()  
 main\_id\_path = os.path.join(path, str(new\_img.get\_id()))  
 full\_path = os.path.join(main\_id\_path, str(this\_version))  
 new\_img.set\_latest\_path(full\_path)  
 new\_img.set\_og\_name('{}{}'.format(new\_img.get\_name(), new\_img.get\_type()))  
 new\_img.set\_full\_name('{}\\{}\_{}{}.{}'.format(full\_path, new\_img.get\_name(), str(new\_img.get\_id()),  
 str(this\_version), new\_img.get\_type()))  
 if os.path.exists(main\_id\_path):  
 if os.path.exists(os.path.join(main\_id\_path, str(this\_version))):  
 image.save(new\_img.get\_full\_name())  
 else:  
 os.mkdir(os.path.join(main\_id\_path, str(this\_version)))  
 image.save(new\_img.get\_full\_name())  
 else:  
 os.mkdir(main\_id\_path)  
 os.mkdir(os.path.join(main\_id\_path, str(this\_version)))  
 image.save(new\_img.get\_full\_name())  
 new\_img.set\_version\_number(new\_img.get\_version\_number() + 1)

**5.5.4 Editing**

העיקר של הקובץ edit\_image הוא פעולות העריכה.

כל פעולה מקבלת כפרמטר תמונה וחלק מהפעולות מקבלות תוספי עריכה נוספים כגון: צבע התמונה, עוצמת העריכה, קורדינטות בשביל החיתוך ועוד.

הפעולות מחזירות את התמונה הערוכה.

**דוגמא לפעולת העריכה שמשנה את צבע התמונה:**

**(הפעולה change\_image\_color)**

def change\_image\_color(image, color, level):  
 *"""  
 changes the image to any color given with 3 different levels.* ***:param*** *image: the image given* ***:param*** *color: the dominant color of the new image. 'red' 'green' 'blue' 'turquoise' 'yellow' 'purple'* ***:param*** *level: the level of selected\_color in the picture. variable must be 1,2 or 3.  
 """* global selected\_color  
 rl = (  
 0.4, 0.4, 0.4, 0,  
 0.1, 0.1, 0.1, 0,  
 0.1, 0.1, 0.1, 0)  
  
 rm = (  
 0.5, 0.5, 0.5, 0,  
 0.1, 0.1, 0.1, 0,  
 0.1, 0.1, 0.1, 0)  
  
 rs = (  
 0.7, 0.7, 0.7, 0,  
 0.1, 0.1, 0.1, 0,  
 0.1, 0.1, 0.1, 0)  
  
 gl = (  
 0.1, 0.1, 0.1, 0,  
 0.3, 0.3, 0.3, 0,  
 0.1, 0.1, 0.1, 0)  
  
 gm = (  
 0.1, 0.1, 0.1, 0,  
 0.5, 0.5, 0.5, 0,  
 0.1, 0.1, 0.1, 0)  
  
 gs = (  
 0.1, 0.1, 0.1, 0,  
 0.7, 0.7, 0.7, 0,  
 0.1, 0.1, 0.1, 0)  
  
 bl = (  
 0.2, 0.2, 0.2, 0,  
 0.2, 0.2, 0.2, 0,  
 0.5, 0.5, 0.5, 0)  
  
 bm = (  
 0.2, 0.2, 0.2, 0,  
 0.2, 0.2, 0.2, 0,  
 0.9, 0.9, 0.9, 0)  
  
 bs = (  
 0.15, 0.15, 0.15, 0,  
 0.15, 0.15, 0.15, 0,  
 0.9, 0.9, 0.9, 0)  
  
 tl = (  
 0.5, 0.5, 0.5, 0,  
 0.6, 0.6, 0.6, 0,  
 0.6, 0.6, 0.6, 0)  
  
 tm = (  
 0.3, 0.3, 0.3, 0,  
 0.6, 0.6, 0.6, 0,  
 0.6, 0.6, 0.6, 0)  
  
 ts = (  
 0.1, 0.1, 0.1, 0,  
 0.6, 0.6, 0.6, 0,  
 0.6, 0.6, 0.6, 0)  
  
 yl = (  
 0.6, 0.6, 0.6, 0,  
 0.6, 0.6, 0.6, 0,  
 0.5, 0.5, 0.5, 0)  
  
 ym = (  
 0.6, 0.6, 0.6, 0,  
 0.6, 0.6, 0.6, 0,  
 0.3, 0.3, 0.3, 0)  
  
 ys = (  
 0.6, 0.6, 0.6, 0,  
 0.6, 0.6, 0.6, 0,  
 0.1, 0.1, 0.1, 0)  
  
 pl = (  
 0.6, 0.6, 0.6, 0,  
 0.5, 0.5, 0.5, 0,  
 0.6, 0.6, 0.6, 0)  
  
 pm = (  
 0.6, 0.6, 0.6, 0,  
 0.3, 0.3, 0.3, 0,  
 0.6, 0.6, 0.6, 0)  
  
 ps = (  
 0.6, 0.6, 0.6, 0,  
 0.1, 0.1, 0.1, 0,  
 0.6, 0.6, 0.6, 0)  
  
 if color == 'red':  
 if level == 1:  
 selected\_color = rl  
 elif level == 2:  
 selected\_color = rm  
 elif level == 3:  
 selected\_color = rs  
 else:  
 print("ERROR. level value can be: 1,2,3")  
 elif color == 'green':  
 if level == 1:  
 selected\_color = gl  
 elif level == 2:  
 selected\_color = gm  
 elif level == 3:  
 selected\_color = gs  
 else:  
 print("ERROR. level value can be: 1,2,3")  
 elif color == 'blue':  
 if level == 1:  
 selected\_color = bl  
 elif level == 2:  
 selected\_color = bm  
 elif level == 3:  
 selected\_color = bs  
 else:  
 print("ERROR. level value can be: 1,2,3")  
 elif color == 'turquoise':  
 if level == 1:  
 selected\_color = tl  
 elif level == 2:  
 selected\_color = tm  
 elif level == 3:  
 selected\_color = ts  
 else:  
 print("ERROR. level value can be: 1,2,3")  
 elif color == 'yellow':  
 if level == 1:  
 selected\_color = yl  
 elif level == 2:  
 selected\_color = ym  
 elif level == 3:  
 selected\_color = ys  
 else:  
 print("ERROR. level value can be: 1,2,3")  
 elif color == 'purple':  
 if level == 1:  
 selected\_color = pl  
 elif level == 2:  
 selected\_color = pm  
 elif level == 3:  
 selected\_color = ps  
 else:  
 print("ERROR. level value can be: 1,2,3")  
 if color == 'red' or color == 'green' or color == 'blue' or color == 'turquoise' or color == 'yellow' \  
 or color == 'purple':  
 updated\_image = image.convert("RGB", selected\_color)  
 save\_to\_dir(updated\_image, None, None)  
 return updated\_image  
 else:  
 if color == 'grey' or color == 'gray':  
 grayscale = image.convert("L")  
 save\_to\_dir(grayscale, None, None)  
 return grayscale  
 else:  
 print("ERROR. color not found. color can be: red, green, blue, turquoise, yellow, purple or grey")

**6. סיכום אישי**

אני מאוד שמח שבחרתי בפרוייקט הזה. למרות שאני בטוח שכל פרוייקט היה משפר את היכולות שלי לתכנת טוב יותר, הפרוייקט הספציפי שבחרתי עניין אותי מאוד ונתן לי כלים רבים לעתיד.

הפרוייקט נתן לי את היכולת להשקיע כמעט חצי שנה, לפחות שעה ביום בדבר ספציפי. זהו אתגר לא פשוט בכלל, שכן רוב הדברים שאנחנו עושים בחיים קצרים מאוד ולהשקיע במשהו תקופת זמן ארוכה כל-כך יכול לייאש ולשעמם מהר מאוד. אני מאמין שהיכולת הזאת שפיתחתי בעזרת הפרוייקט תבוא כעת לידי ביטוי בדברים נוספים שאעשה בחיים שלאו דווקא קשורים לתכנות.

בנוסף, הפרוייקט שיפר את רמת התכנות שלי משמעותית.

בכיתה יא' כאשר ניגשתי לבגרות במחשבים, הייתי משוכנע שאני יודע טוב מאוד לתכנת (גם הציון של הבגרות חיזק את ההנחה שהייתה לי). אך רק עכשיו אני מבין כמה גדול התחום הזה, ושאני רק טעמתי טעימה קטנה ממנו. נהנתי מאוד לתכנת את הפרוייקט. לעיתים חיכיתי שאגיע הביתה כי כבר היה לי בראש מה אני צריך לשנות בקוד בשביל שהבאג יפתר. וכל פעם שעברתי עוד מסוכה ההרגשה הייתה עילאית.

הפרוייקט בפרט והמגמה בכלל היוותה יציאה מאזור הנוחות שלי.   
לעיתים אני רק רציתי להנות מהזמן החופשי ולהיפגש עם חברים, אך לא הייתי משנה דבר וזוהי המגמה שהעשירה את הידע שלי בצורה הגדולה ביותר.

**7. ביבליוגרפיה**

PILLOW DOCUMENTATION:  
  
<https://pillow.readthedocs.io/en/stable/>

SOCKETS:

<https://docs.python.org/3/howto/sockets.html>

<https://realpython.com/python-sockets/>

<https://realpython.com/python-print/>

IMAGES:

<https://en.wikipedia.org/wiki/Digital_image>

GUI:

<https://www.youtube.com/watch?v=YXPyB4XeYLA>

<https://realpython.com/python-gui-tkinter/>