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


P.E.P
PHOTO EDIT PLATFORM
OMER OZ

תוכן עניינים –

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.

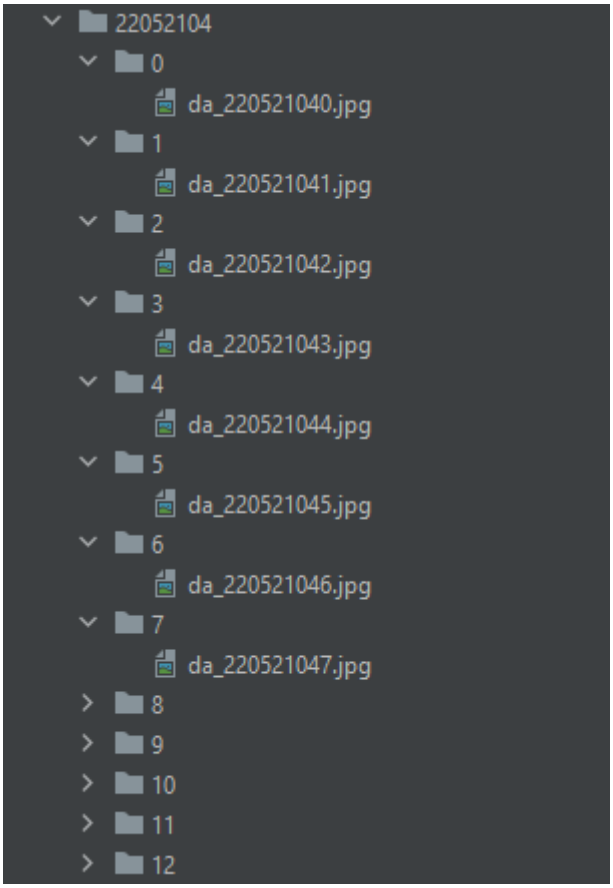
1.6 בעיות בכתיבת הפרוייקט ופתרונות -

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 -

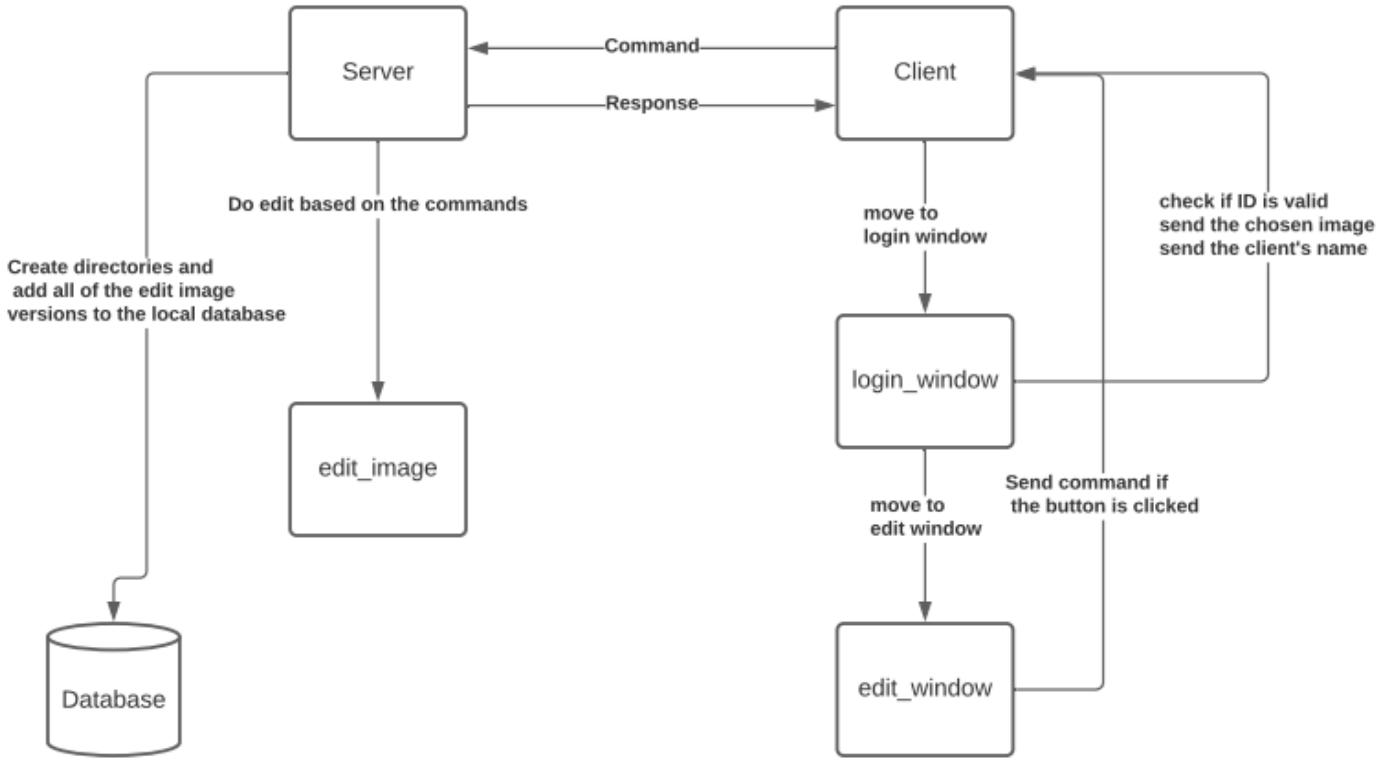
עריכת התמונות היא חלק בלתי נפרד מהחיים שלנו. התוכנה יכולה להתאים במיוחד לקבוצות\חברות אשר צריכות לערוך ביחד תמונה ספציפית. שכן הדבר יקל עליהם ויחסוך זמן יקר.

2. 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

Front-End 2.1

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

Login Window 2.1.1

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 ה.ID
במידה והID לא ואלידי, השרת יחזיר ללקוח תשובה שלילית ויופיע במסך בקשה להכניס קוד ID חדש (ראה תמונה 4)

תמונה 3

תמונה4

Edit Window 2.1.2

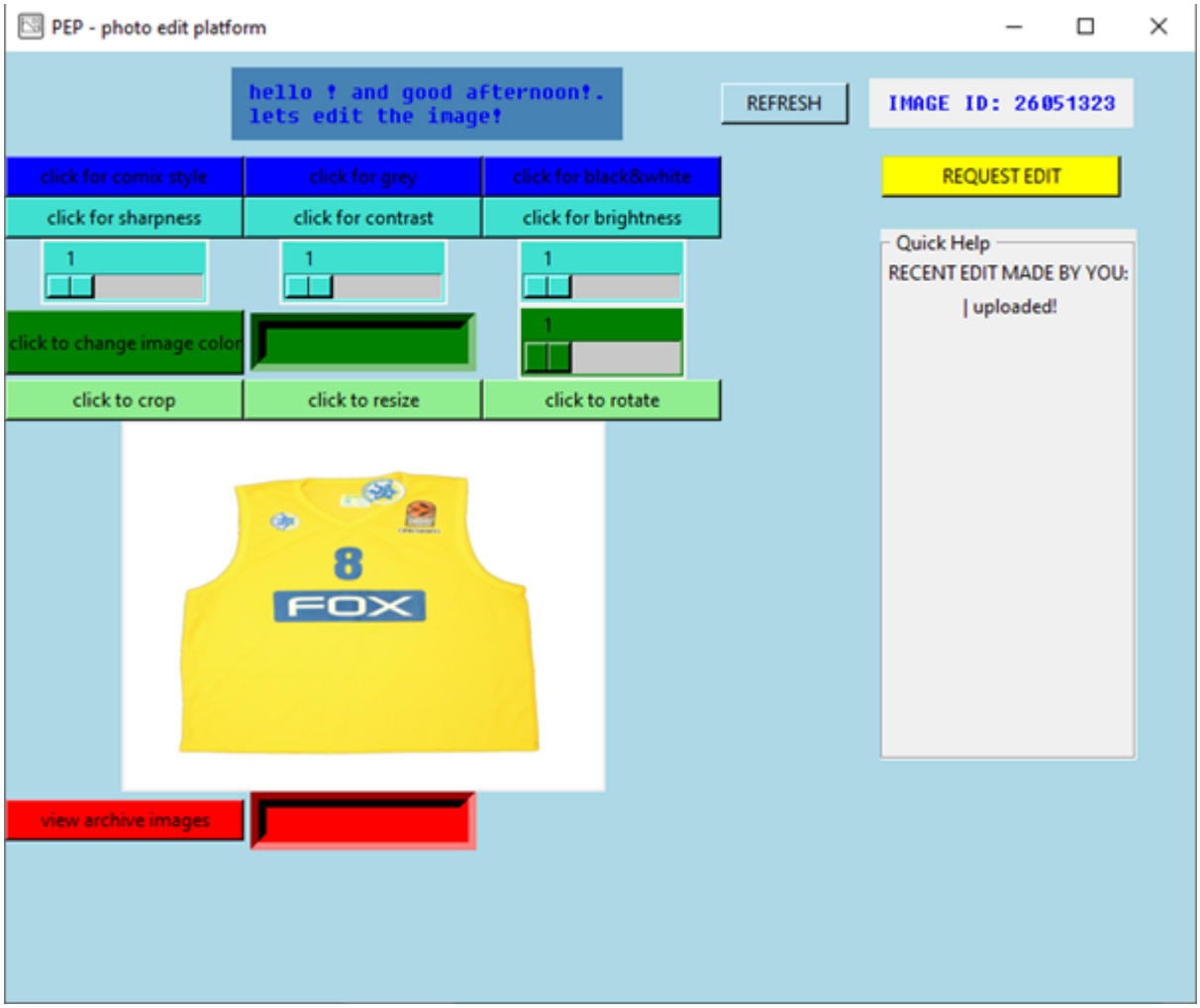
בחלון העריכה מוצג כפתור ריפרש, בקשת עריכה, ID של התמונה, כפתורי העריכה, התמונה עצמה וכפתור לצפות בתמונות בארכיון (ראה תמונה 5).

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

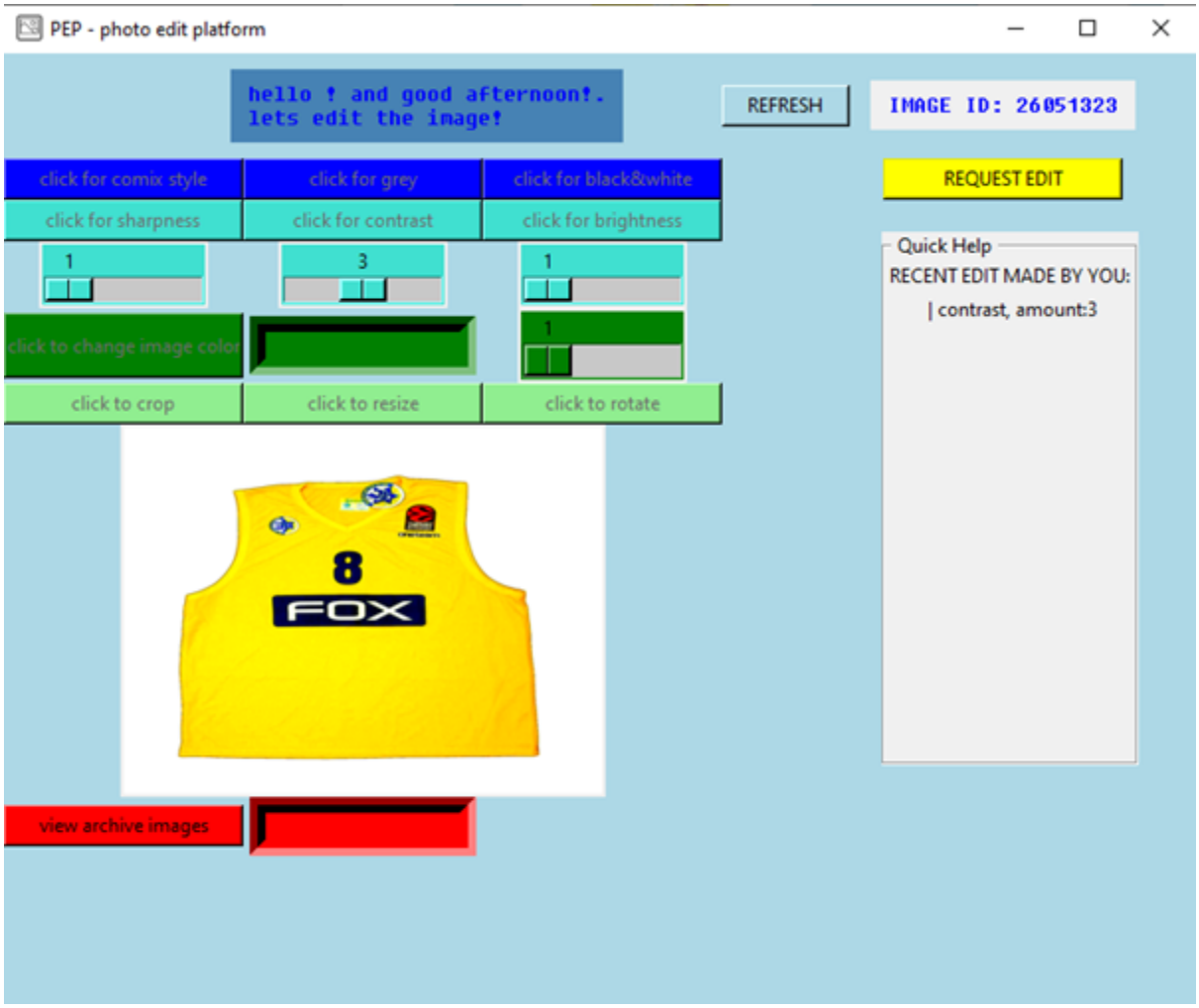
<u>כפתורי העריכה הזמינים למשתמש הם:</u>

- 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



תמונה6

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#

Back-End 2.2

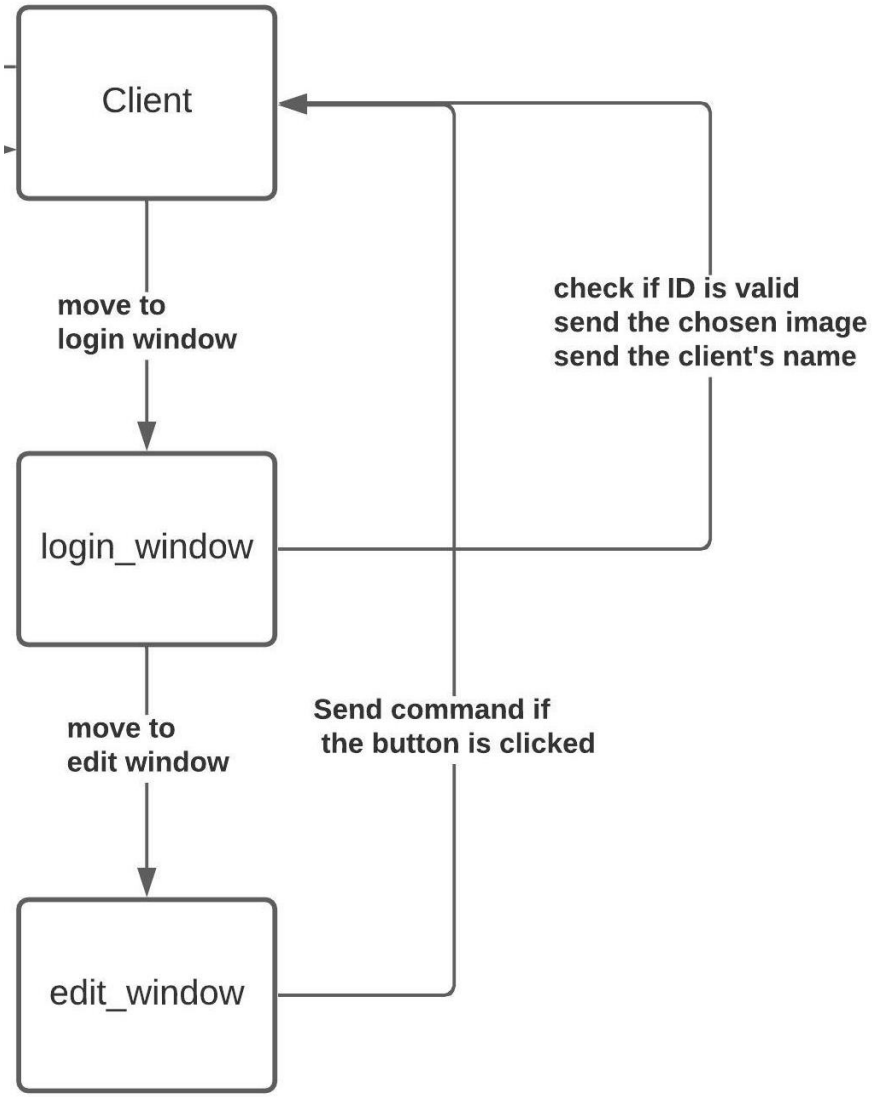
חלק זה מכיל את הלקוח, השרת וקובץ העריכה edit_image.
הלקוח לא נמצא באינטרקציה עם קבצים אלו.

Client 2.2.1

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"

Server 2.2.2

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)

edit_image 2.2.3

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.


gallery.ico

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

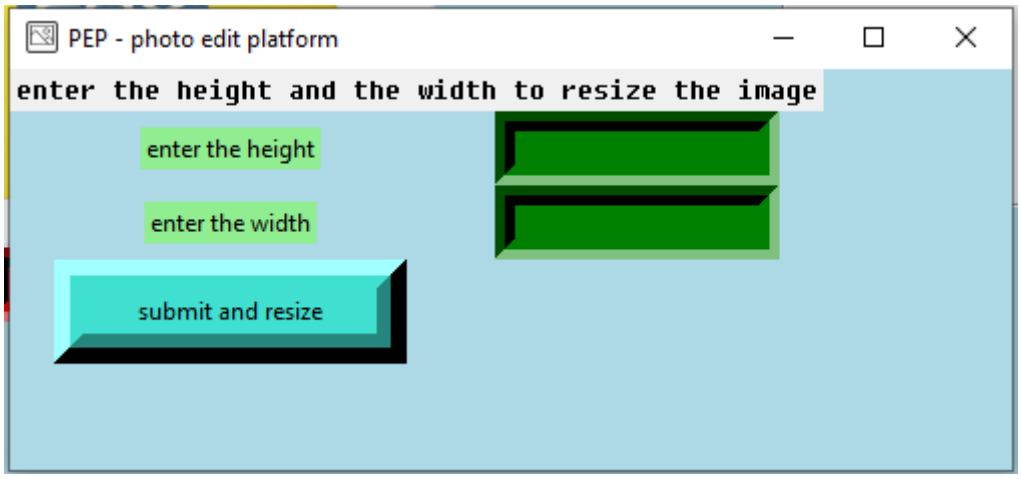
תפקיד מסך ה login window הוא לאפשר למשתמש להעלות תמונה \ להכניס ID בשביל לצפות בתמונה קיימת ולערוך אותה.

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 -

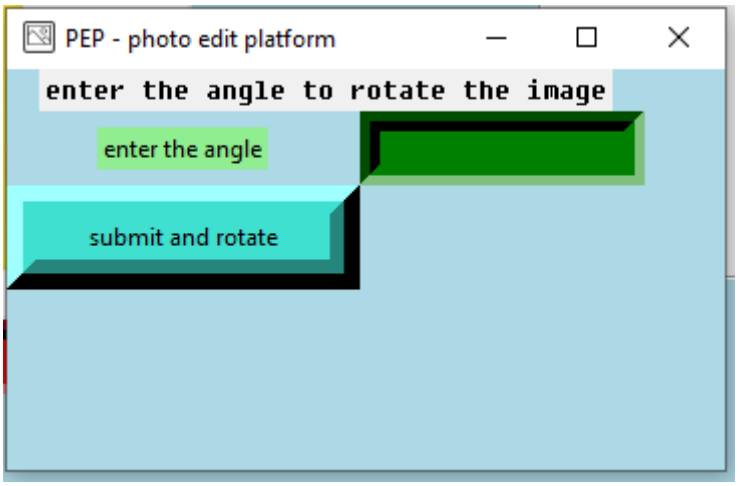
Resize Window



תמונה 7

חלון זה מאפשר לשנות את גודל התמונה

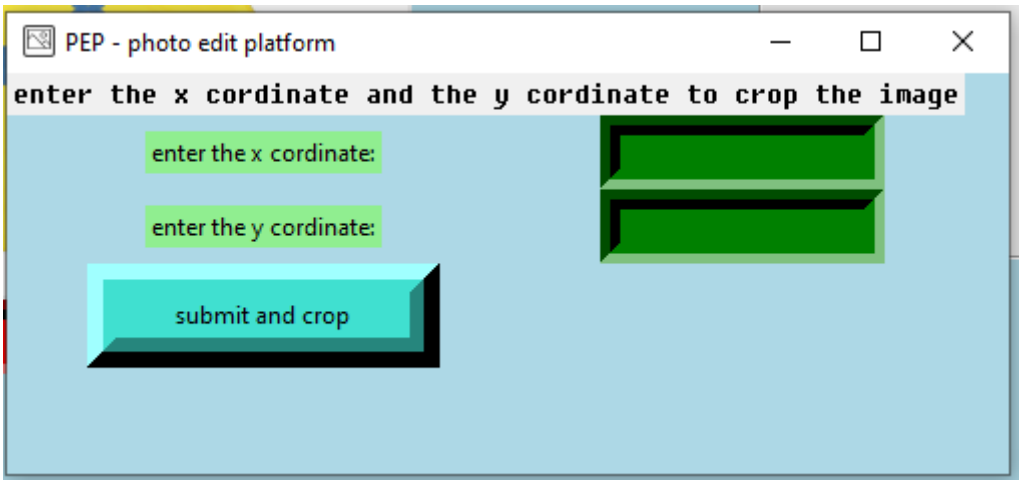
Rotate Window



תמונה 8

חלון זה מאפשר להטות את התמונה.

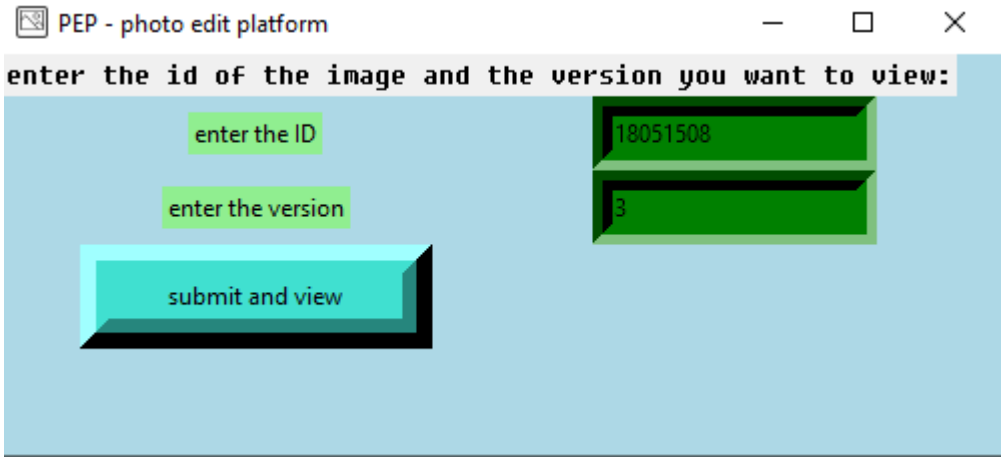
Crop Window



תמונה 9

חלון זה מאפשר לחתוך את התמונה

Archive window



תמונה 10

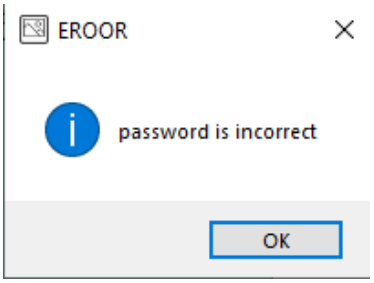
חלון זה מאפשר לצפות בתמונה מארכיון התמונות של בסיס הנתונים

תמונה - 11 התמונה הערוכה המתקבלת לאחר בקשת התמונה מהארכיון

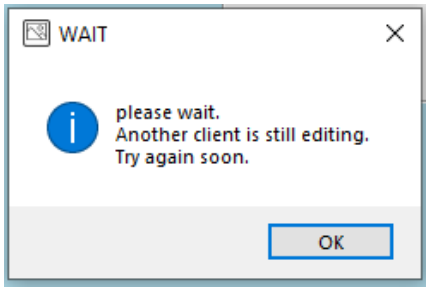
3.4 הסבר הודעות למשתמש

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



תמונה13 - במידה והסיסמא הדרושה לצפייה בתמונות מהארכיון לא נכונה, יופיע החלון הקופץ הנל

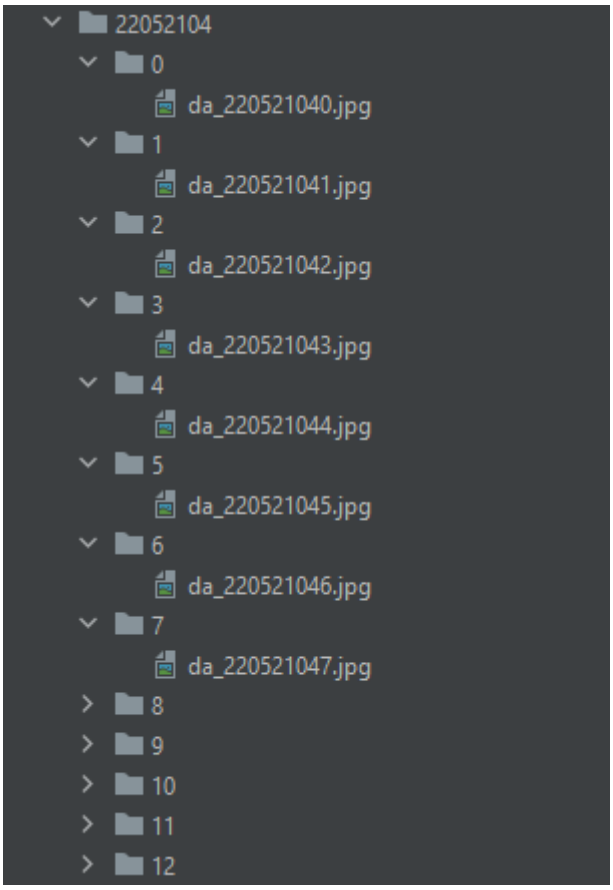


תמונה14 - במידה ולקוח לוחץ על כפתור בקשת העריכה בזמן שלקוח אחר עדיין עורך את התמונה, יופיע החלון הקופץ הנל

4. בסיס נתונים

4.1 אופן הפעולה והשמירה בבסיס הנתונים

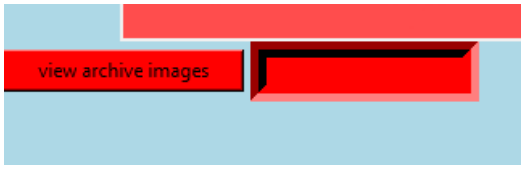
שמירת התמונה בבסיס הנתונים נעשית בצד של השרת.
לכל לקוח אשר יוצר חיבור עם השרת ID ייחודי משלו, וכל גרסה של תמונה תקבל מספר. בבסיס הנתונים קיים לכל ID חדש תיקייה, ותיקיות נוספות בתוכה לכל גרסה של התמונה.
כך כל התמונות נשמרו באופן מסודר (ראה תמונה 15).



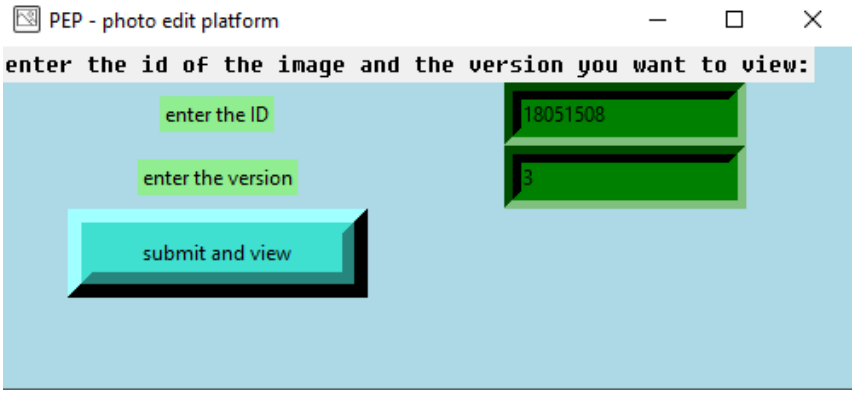
תמונה 15 - דוגמא תיקייה בבסיס הנתונים

4.2 שליפת תמונות מבסיס הנתונים(archive 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).



תמונה16



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.

send_request_to_server 5.1.4

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” #

<u>FOR EXAMPLE –</u>

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())
```

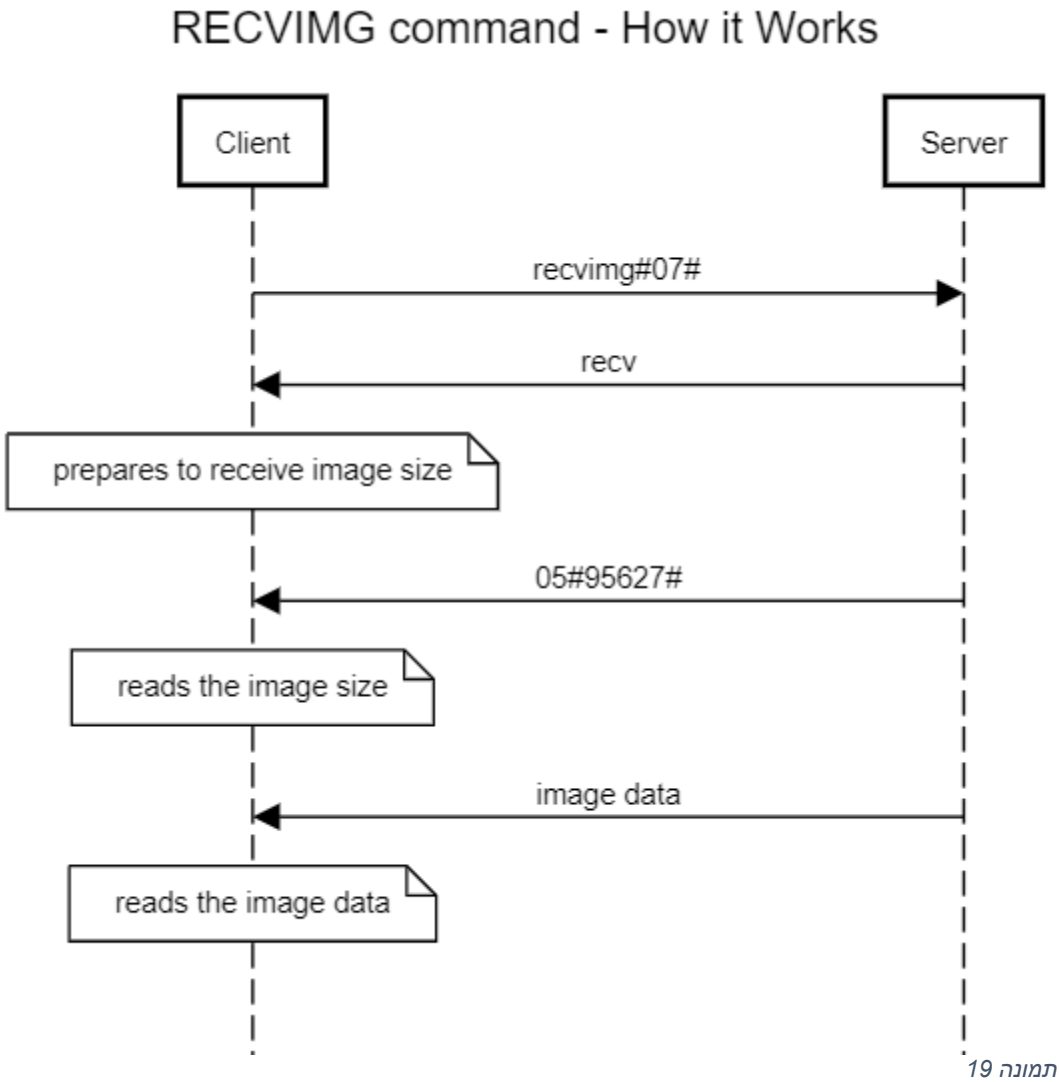
receive_server_response 5.1.5

הפעולה מקבלת מילה בעלת 4 אותיות מהשרת
התגובות האפשריות לקל מהשרת הן:

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

handle_server_response 5.1.6

פעולה זו מבצעת פעולה כלשהי בהתאם לresponse.
לדוגמא כאשר ה"recv" is response, 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)
```

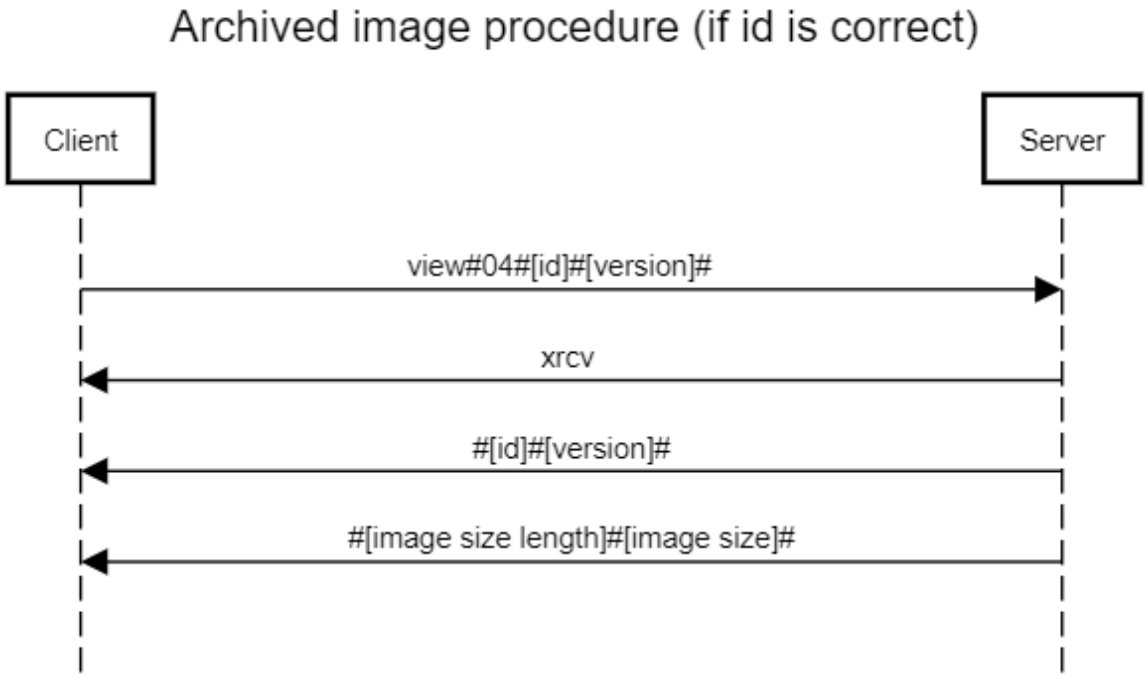
5.3 המחלקה edit_window

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

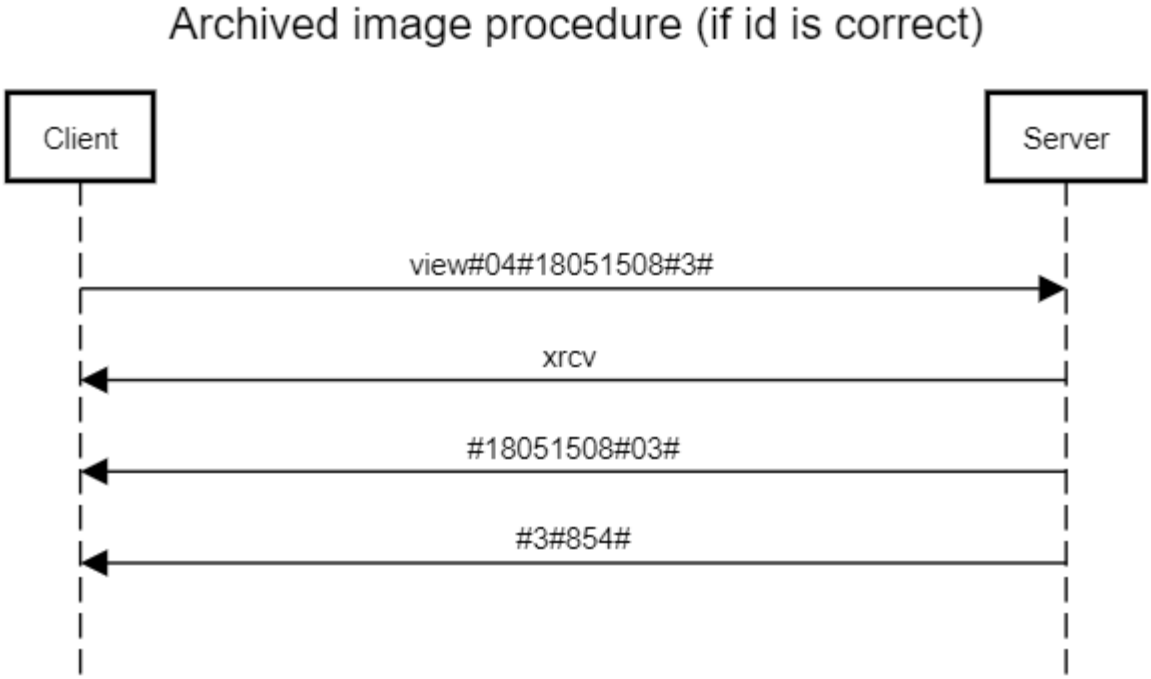
5.3.1 archive_image

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):

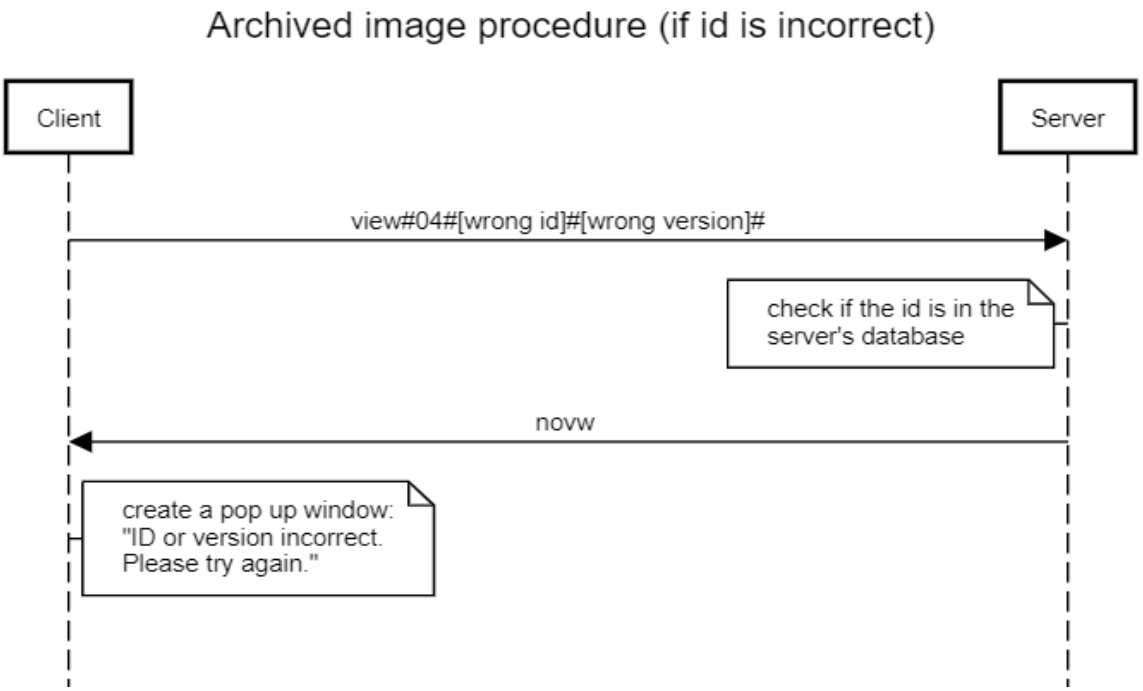


תמונה 20

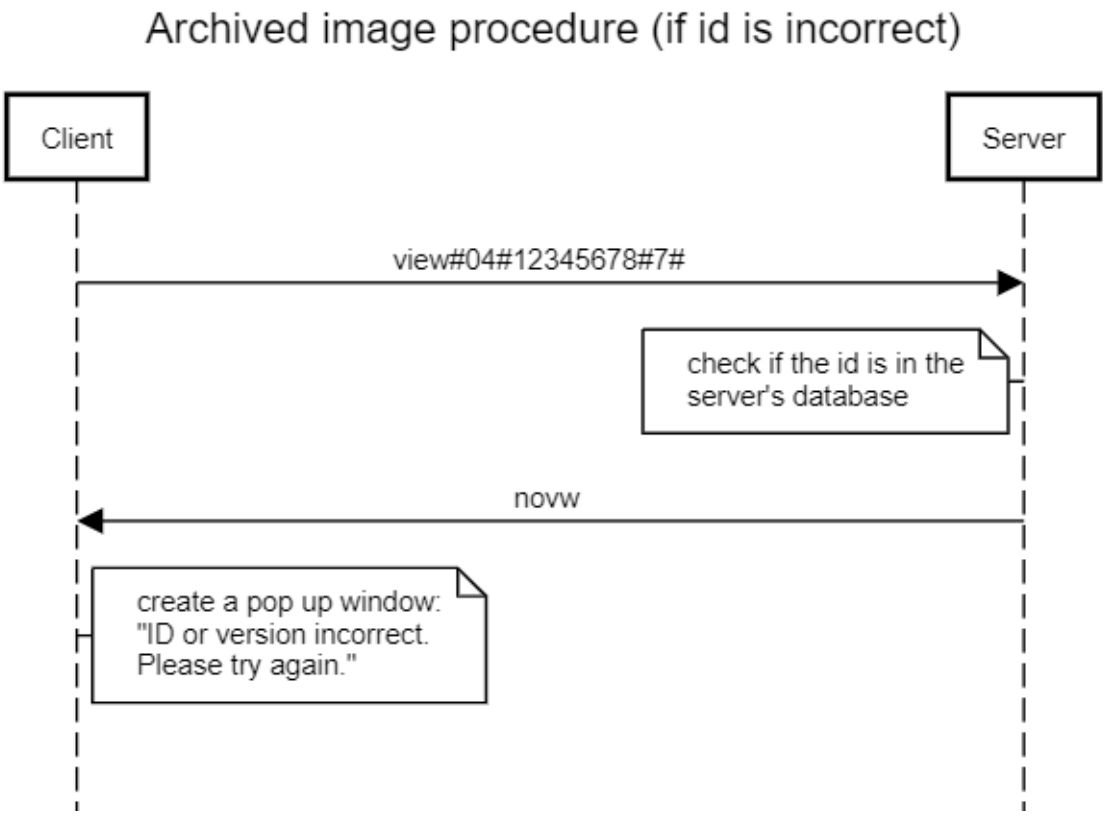


תמונה 21

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



תמונה 22



תמונה 23

view_image 5.3.1.1

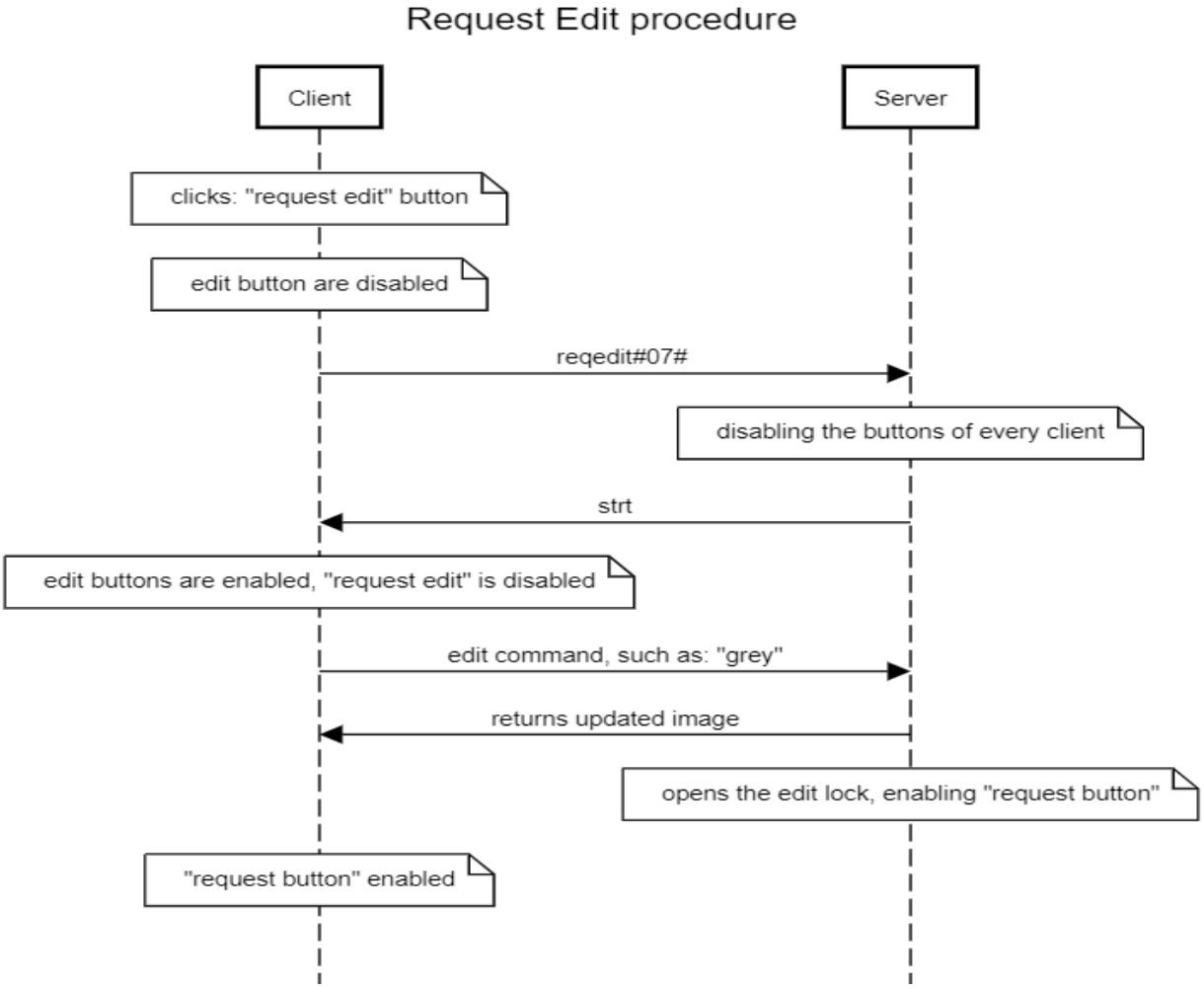
פעולה זו אחראית על פתיחת חלון חדש שבו תוצג התמונה מבסיס הנתונים. בחלון זה תופיע התמונה וכפתור יציאה לחזרה למסך העריכה.

הפונקציה view_image

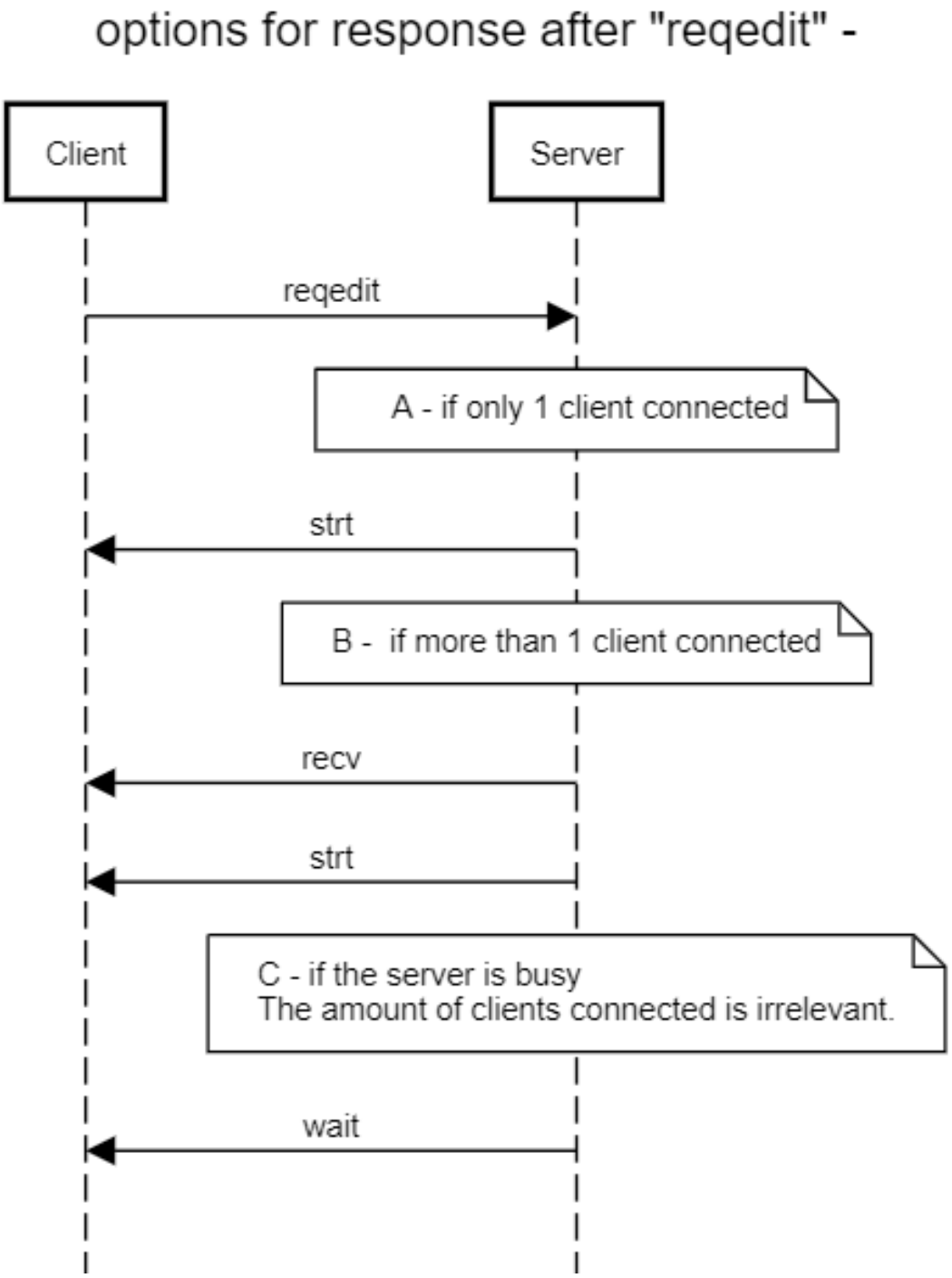
```
def view_image(self, password):
    """
    creates a popup window to rotate the image
    """

    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")
```

request_edit 5.3.2

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

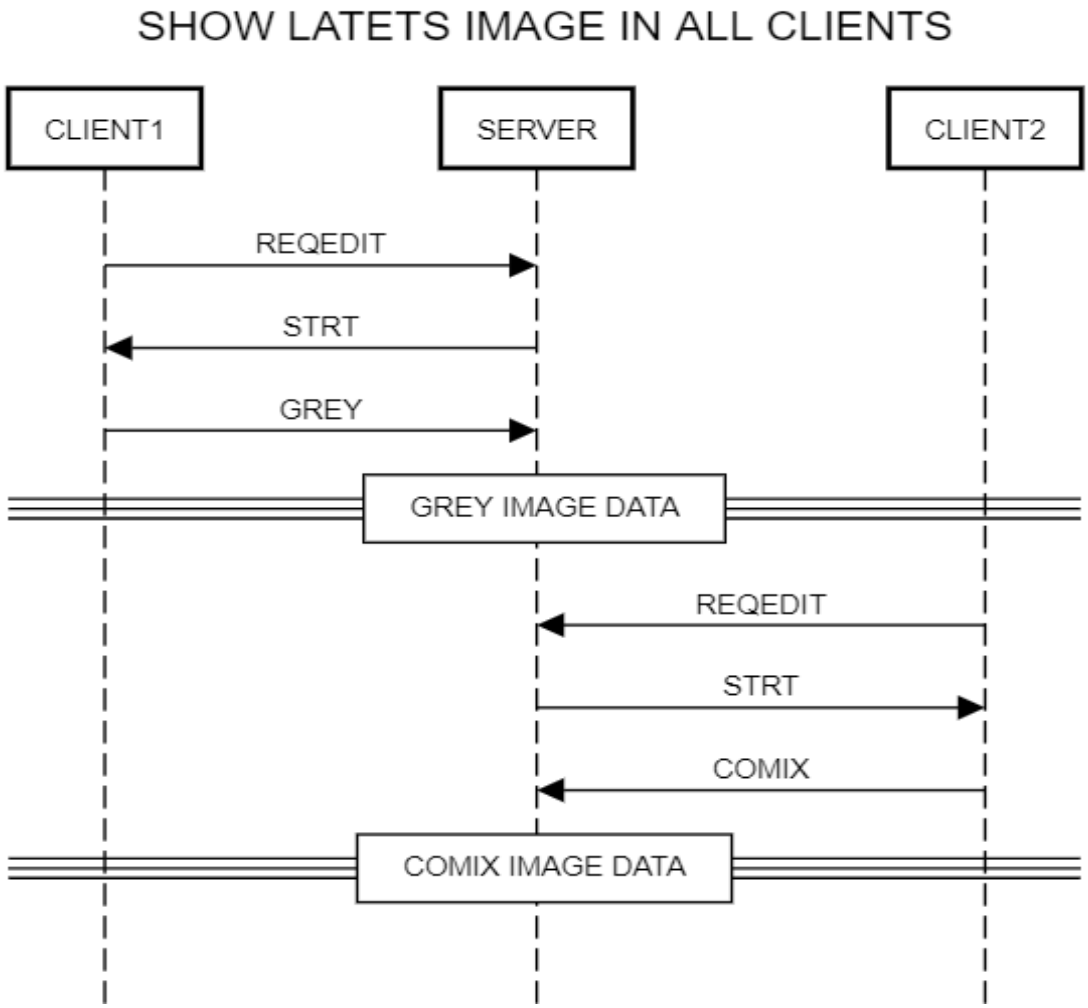
הפעולה do_edit

```
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
```

update_image 5.3.3

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:



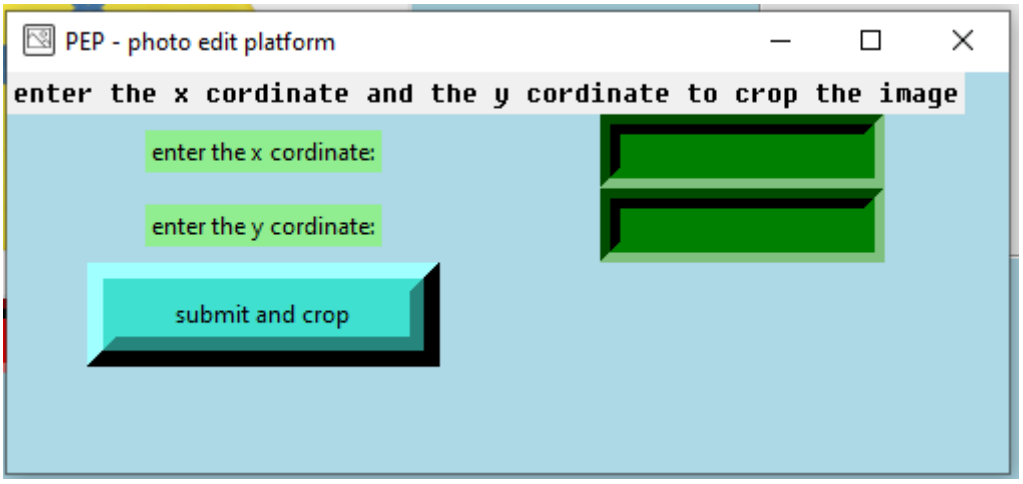
תמונה 26

edit_buttons 5.3.4

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.

win_crop 5.3.4.1

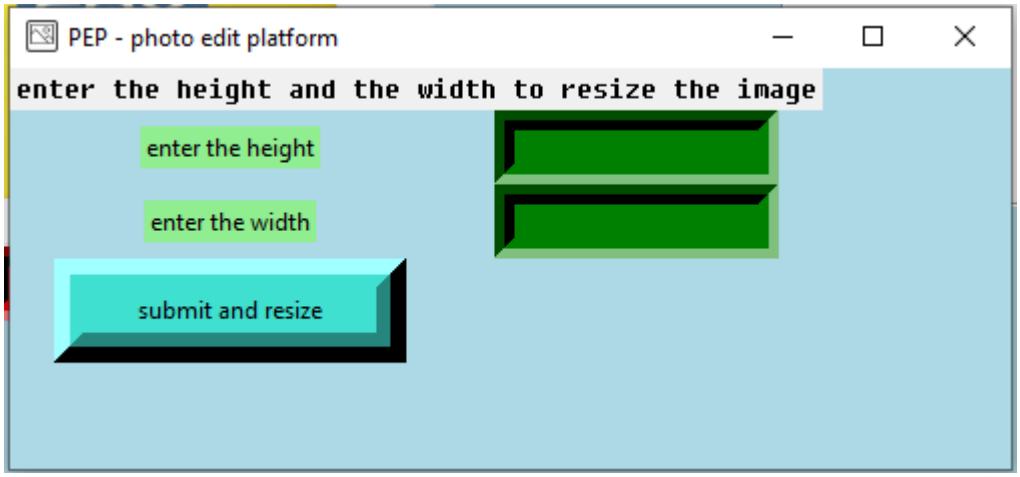
חלון אשר נפתח בעת לחיצה על כפתור החיתוך.
נועד להפוך את העריכה לידידותית יותר.
כאשר המשתמש לוחץ על אישור העריכה, נשלח לשרת בקשת העריכה והנתונים הנדרשים בשביל העריכה הספציפית.



תמונה 27

win_resize 5.3.4.2

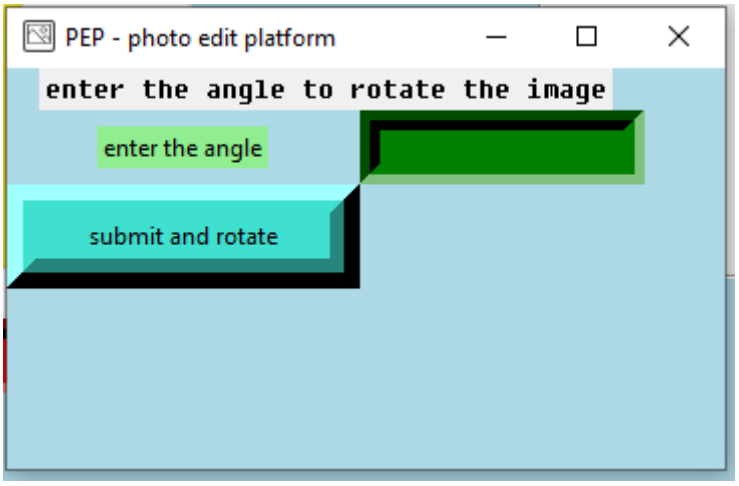
חלון אשר נפתח בעת לחיצה על כפתור החיתוך.
נועד להפוך את העריכה לידידותית יותר.
כאשר המשתמש לוחץ על אישור העריכה, נשלח לשרת בקשת העריכה והנתונים הנדרשים בשביל העריכה הספציפית.



תמונה 28

win_rotate 5.3.4.3

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

Server 5.4

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 וset 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: Q_DATA.

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 ב thread 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 "reqedit" "sendimg" ומחזירה מילה בעלת 4 אותיות. הטיפול בשאר הבקשות יעשה בפעולה הבאה: send_response_to_client

send_response_to_client 5.4.5.3

הפעולה מטפלת בפקודות "id", "vid", "view", "recv" 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.

handler_thread 5.4.6

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.

receive_client_request 5.4.6.1

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
```

handle_image_edit 5.4.7

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

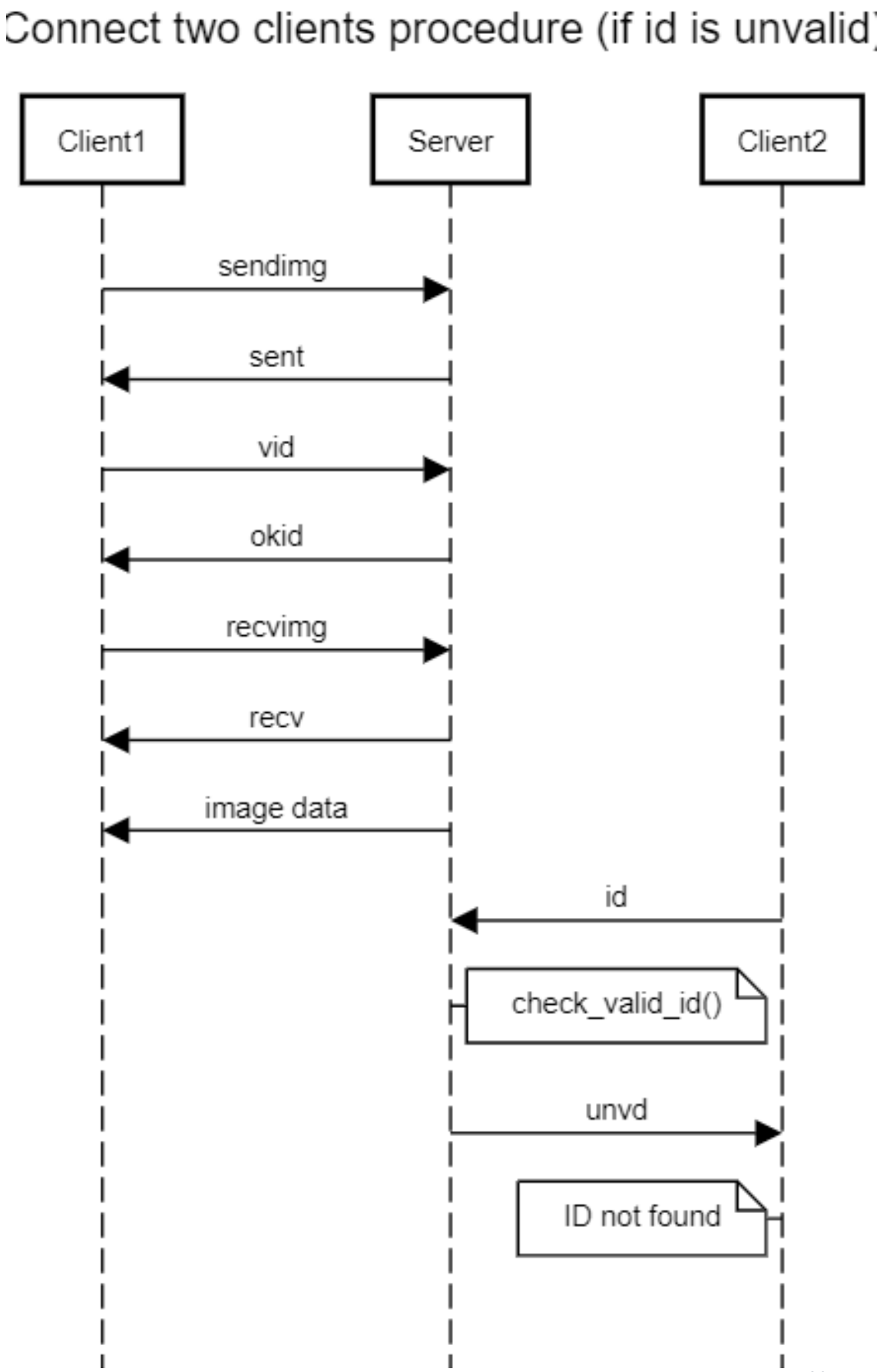
פעולה זו הופכת את התמונה שהתקבלה כפרמטר לפורמט של בתים בכדי שיהיה אפשר לשלוח אותם.
לאחר מכן הפעולה שולחת ללקוח את גודל התמונה ואורך הגודל שלה.
הפעולה שולחת את המידע של התמונה ללקוח ומעדכנת את המחלקה Busy שהעריכה הושלמה וכעת לקוחות אחרים יכולים לערוך את התמונה.

```
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!")
```

check_valid_id 5.4.9

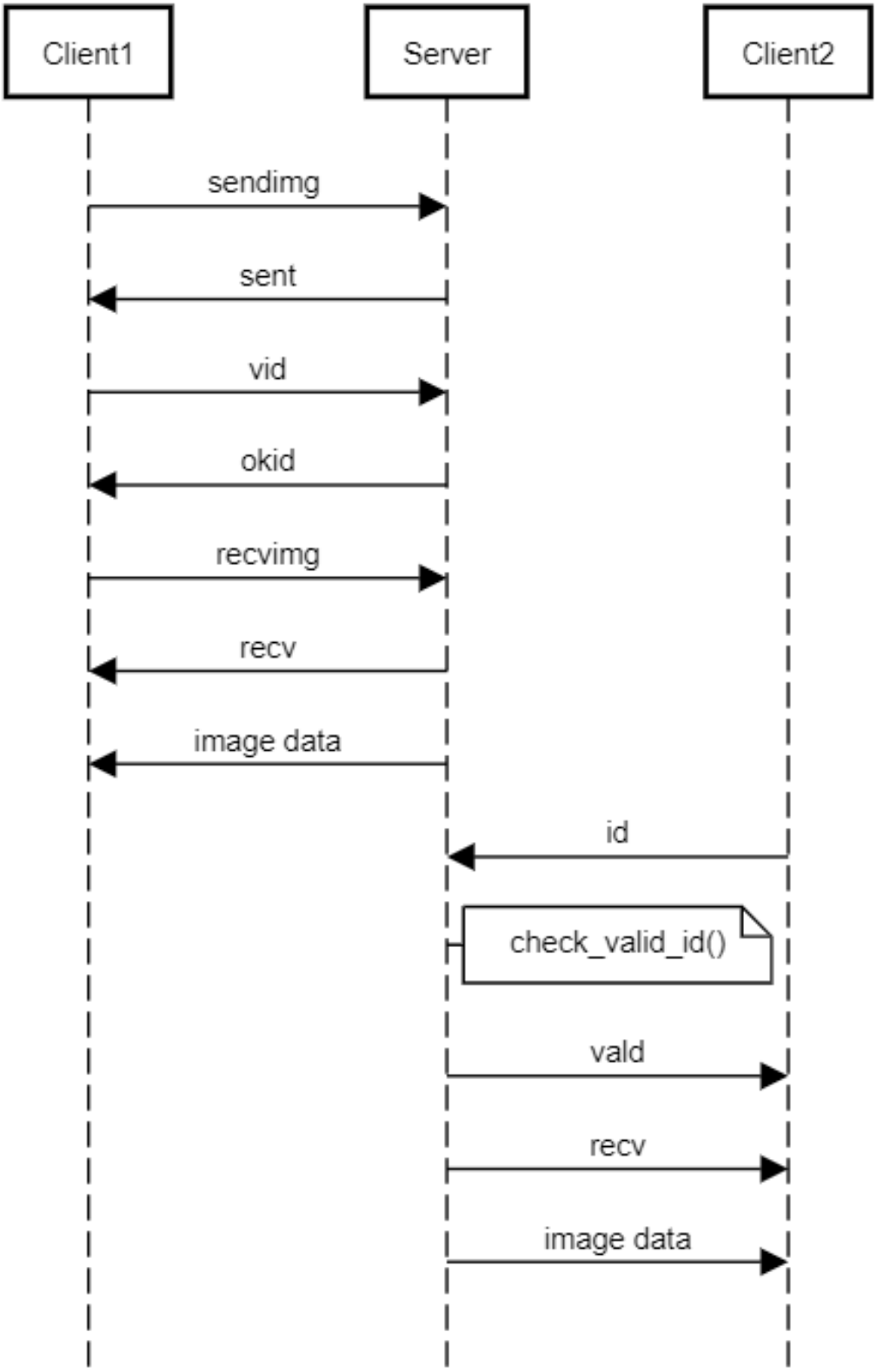
כאשר הלקוח הכניס ID בחלון הכניסה, הוא שולח פרוטוקול של המילה "id" ושל ה ID שהכניס. כאשר השרת מקבל את הפקודה "id", הוא בודק אם ה id תקין על ידי הפעולה check_valid_request.
במידה וה ID נמצא ברשימת ה id_list, השרת יחזיר אישור, את הפקודה"vald" וישלח ללקוח את התמונה העדכנית ביותר בשרת.

במידה וה ID לא נמצא ברשימת ה id_list, השרת יחזיר את הפקודה "unvd" שאומרת שה ID לא תקין.


Connect two clients procedure (if id is unvalid)
Client1
Server
Client2
sendimg
sent
vid
okid
recvimg
recv
image data
id
check_valid_id()
unvd
ID not found

תמונה 30

Connect two clients procedure (if id is valid)



edit_image 5.5

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

read_image 5.5.2

הפעולה קוראת את התמונה ומעבירה אותה לסוג PIL.

save_to_dir 5.5.3

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)
```

Editing 5.5.4

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/