



## מערכת מבוזרת לזיהוי מבוקשים (בעזרת זיהוי פנים) Global-Watch "תמונה שווה אלף מילים"



שם הפרויקט: Global Watch  
מגיש: דביר פרחיה  
תאריך: 6/6/2023

---

שם התלמיד: דביר פרחיה

מספר תעודת זהות: 329194757

מורה: יורם

תאריך הגשה: 6/6/2023

## Global-Watch

### תוכן עניינים

#### תוכן עניינים

4.....	טבלת שינויים לאורך ממימוש הפרויקט	
5.....	1 מבוא – תיאור נושא הפרויקט	
6.....	1.1 נושא המחקר בפרויקט	
7.....	2 סביבת העבודה והספריות העיקריות שבשימוש בפרויקט	
8.....	2.1 טכנולוגיות בשימוש בפרויקט	
8.....	2.2 מדריך למשתמש	
9.....	3 אפיון דרישות וארכיטקטורת מערכת	
9.....	3.1 דרישות ושימושי מערכת – USE CASES	
9.....	3.2 סביבת הפרויקט – ECO – SYSTEM	
10.....	3.3 ארכיטקטורת המערכת	
13.....	3.4 טבלת בדיקה עבור ארכיטקטורה ודרישות מערכת	
14.....	4 ממשק משתמש - GUI	
24.....	5 מדריך למפתח	
24.....	5.1 דיאגרמת UML של כל מחלקות הפרויקט והתלויות ביניהן	
25.....	5.2 רשימת פונקציות ומחלקות ותפקידיהם	
28.....	6 סיכום אישי / רפלקציה	
30.....	7 מקורות מידע / ביבליוגרפיה	
31.....	נספח	

## טבלת שינויים לאורך ממימוש הפרויקט

תאריך סיום	תכולה / שינוי	גרסה	פעילות
20/10/22	בחירת הנושא לפרויקט	0.1	יזום
10/11/22	תחילת עבודה על ספר הפרויקט, מחקר על מה נדרש בשביל לממשו ותחילת כתיבת ממשק משתמש רעיוני	0.2	אפיון דרישות, ארכיטקטורת מערכת, כלי פיתוח
5/1/23	ממשק משתמש מוכן ותכנון המערכת	1	קידוד v1.0 שלד עובד
2/2/23	הוספת מאגר נתונים ותחילת בניית פרוטוקול תקשורת	2	קידוד v2.0
2/3/2023	סיום בניית פרוטוקול התקשורת ומימוש	3	קידוד v3.0
27/3/23	הוספת זיהוי פנים בלקוח	4	קידוד v4.0

## 1 מבוא – תיאור נושא הפרויקט

הפרויקט שלי עוסק בתחום הזיהוי הביומטרי - בזיהוי פנים. הזיהוי ביומטרי הוא שם לכלל השיטות המשמשות לזיהוי בני אדם אשר מבוססות על זיהוי של תכונות פיזיות או התנהגותיות.

### מדוע צריך זיהוי פנים?

בתחילת הדרך של הזיהוי הביומטרי נעשה שימוש נרחב במערכות ביומטריות כגון זיהוי טביעת אצבע ולאחר מכן זיהוי קשתית העין אך אלו אמצעים הדורשים את שיתוף הפעולה וידעתו של המזדהה. בתמונה למטה ניתן לראות דוגמא לזיהוי של קשתית העין:



הצורך במערכות זיהוי פנים נולד בעקבות הצורך לזיהוי ביומטרי ללא ידיעת האדם המזוהה על ידי רשויות אכיפת החוק ברחבי העולם. בהמשך הדרך הפכו אמצעי זיהוי הפנים לאמצעי זיהוי ביומטרי להמונים שלא מצריך את תשומת לב המזוהה אלא מבצע את הזיהוי והגילוי בדרך אגב ולעיתים ללא ידיעת המצולם.

### כיצד זיהוי הפנים עובד?

מערכות לזיהוי תווי פנים הינן בדרך כלל אפליקציות מבוססות מחשב אשר מסוגלות לזהות ולאמת את זהותו של אדם באופן אוטונומי על בסיס תמונה של אותו אדם שהוזנה מראש למערכת.

ישנן מספר שיטות לגילוי וזיהוי המטרה. רוב האפליקציות של מערכות זיהוי הפנים משתמשות בטכנולוגיות המשוות את תכונות תווי הפנים שהורכשו ע"י המערכת לאלה הנמצאות במאגר (והוזנו לשם מראש).

מערכות לגילוי וזיהוי תווי פנים משתמשות במגוון רחב של אלגוריתמים ושיטות זיהוי

לדוגמא, **זיהוי תווי פנים דו-ממדי**. האלגוריתם הנ"ל מבצע אלימינציה של תכונות שאינן משתתפות בתהליך הזיהוי. האלגוריתם ינסה לבדוק את הגודל והמיקום היחסי (בפנים של האדם הנבדק) של העיניים, או את המרחק בין האוזניים וכדומה או שילוב בין כמה מדידות.

המדידות הללו יושוו לאחר מכן למדידות שבוצעו מראש על מאגר הנתונים הקיים וכך יתבצע הזיהוי או שלילת הזיהוי.

שיטה נוספת לזיהוי פנים היא **ניתוח טקסטורת העור**. השיטה הזו בודקת ומשווה את מרקם העור ורוב הפעמים לא משתמשים בה כשיטה העיקרית אלא משומשת כנספח לשיטת זיהוי אחרת מרכזית כדי להגביר את התוצאות לרמת דיוק גבוהה יותר.

אלגוריתם נוסף הוא **זיהוי תווי פנים תלת ממדי**. באלגוריתמים מהסוג הזה משתמשים בכמה חיישנים תלת ממדיים על מנת לבצע את הרכשת המטרה ולאסוף נתונים כגון גודל הפנים, גודלן של העיניים ושל איברים אחרים ואז כמו בשיטות אחרות להשוות את המדידות לאלו שבוצעו על תמונות המאגר. מוכח שטכניקה זו משיגה רמת דיוק גבוהה יותר משיטות אחרות דו ממדיות.

מוצרים העושים שימוש בזיהוי פנים נמצאים כמעט בכל מקום בעולם האבטחה החל מזיהוי המונים בשדה תעופה, עבור בזיהוי מבוקשים בין קהל רב ובקרת כניסה.

בפרויקט שלי אשתמש באלגוריתם זיהוי הפנים הדו ממדי ואבנה מערכת שהיא מערכת של שרת לקוח שתבצע זיהוי פנים.

אבנה מנגנון זיהוי פנים של מבוקשים. לשרת יהיה מאגר של פרצופים של אנשים מבוקשים והלקוח יהיה מצלמה(כמו מצלמת רחוב, בחנות, בקניון) אשר מצלמת ווידאו וברגע שהיא מזהה מבוקש היא תעדכן את השרת שזיהתה מבוקש והשרת יוסיף את המבוקש על המפה בעמוד הראשי שלו. וכך תיווצר מפה שעליה ניתן לראות איפה המבוקשים בעולם.

## 1.1 נושא המחקר בפרויקט

בפרויקט שלי אחקור על טכנולוגיית זיהוי הפנים ופרט על אלגוריתמים יעילים לזיהוי פנים ואנסה לממשם בפרויקט שלי.

קיימות בעיות רבות בזיהוי הפנים ובאגים שעם מנצלים ניתן "לעבוד" על המערכת כגון: האקרים הגונבים פנים של אנשים כדי להתחבר למקומות מסוימים, הדפסה של הפרצוף של אדם ללא צורך באדם עצמו, או לקיחת תמונה שלו ולהשתמש בה לאחר מכן שימוש בה להתחברות ועשיית זדון. בעיה נוספת היא בעיית הפרטיות –ניתן לעקוב אחרי אנשים בעזרת זיהוי הפנים שלהם וזה מהווה חדירה לפרטיות. לכן, חשוב להמשיך לחקור ולקדם את תחום זיהוי הפנים בגלל כל הבעיות המצויות בו כי הוא מכיל בתוכו פוטנציאל עצום בתחום הזיהוי הביומטרי. בפרויקט שלי אצור מערכת מבוססת לזיהוי מבוקשים בעזרת זיהוי פנים.

בחרתי בנושא זה מכיוון שמעניין אותי הנושא של זיהוי פנים ולבנות מערכת שתזהה מבוקשים בעזרת רשת של מצלמות ברחבי העולם נשמע רעיון מסקרן(וגם בהרבה סרטים מופיעים דברים דומים) ועניין אותי לחקור על הנושא. בפרויקט זה אבנה מערכת של שרת לקוח שבו השרת מחזיק את מאגר הנתונים של התמונות, שולח אותו ללקוח- מצלמה והוא מבצע את הזיהוי ומעדכן את השרת אם זיהה מבוקש.

כיום משתמשים בהרבה מקומות בטכנולוגית זיהוי, מקומות כגון: מקומות עבודה (בכניסה לבניין או למתחם כלשהו ואפילו בזיהוי למחשב)

בנקים

בתי חולים

קזינו

שדות תעופה

טלפונים ניידים

לצורכי המשטרה או הצבא

ואף יש חברות פרטיות שמוכרות מתקנים המשתמשים בזיהוי הפנים שניתן להתקין אותם בדלת  
בייתנו ולהפיכתו לחכם. דוגמאות למכשירים הנמכרים כיום:

## Featured Products



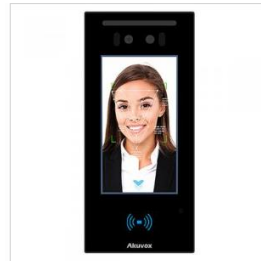
Akuvox R29C - Smart IP Intercom with face recognition & QR Code reader - silver

[view product](#) +



Akuvox R29CBLK - Smart IP Intercom with face recognition & QR Code reader - black

[view product](#) +



Akuvox A05 Access Unit with face recognition, Bluetooth, RFID & QR Code

[view product](#) +



Akuvox - X915S IP Touchscreen Smart Door Intercom Unit with Face Recognition, QR, RFID, BLE

[view product](#) +

## 2 סביבת העבודה והספריות העיקריות שבשימוש בפרויקט

אני אשתמש בשפת התכנות פייתון גרסה 3.8 עם 64 ביט (בחרתי בפייתון כי יש בה הרבה ספריות מובנות ולכן אני לא אצטרך לממש הכול מ0).

3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:37:50) [MSC v.1916 64 bit (AMD64)]

כמה ספריות עיקריות:

1. Open cv - סיפריה שאחראית על קריאת ווידאו ממצלמה
  2. Facial\_recognition ספרייה שמבצעת זיהוי פנים
  3. Kivymd, kivy - בשביל guin.
  4. Sqlite3 - ספרייה שבעזרתה אוכל לגשת ולכתוב למאגר הנתונים
  5. SSL - ספרייה שבאמצעותה נעשה חיבור TLS דרך הסוקט
  6. Socket - ספרייה זו מאפשרת פתיחת סוקטים ושליחת מידע על גבי האינטרנט
  7. Threading - ספרייה שבעזרתה ניתן לפתוח תהליכים
  8. Pickle - בשביל שליחת מאגרי מידע ומידע רגיל דרך הסוקט
  9. Os - להפסקת ההרצה ומחיקת קבצים
  10. Hashlib - בשביל להצפין את הסיסמאות במאגר נתונים.
  11. Time - בשביל ליצור השהיה
  12. Numpy - לעבודה עם מערכים שהזיהוי פנים עושה בהם שימוש.
  13. Captcha\_image - בשביל ליצור את תמונת ה captcha
  14. Random - בשביל ליצור ID רנדומלי ואת הטקסט של ה captchan
- Pytttsx3 - בשביל שהמערכת תגיד את השמות של המבוקשים שזוהו.

15. Geopy על מנת מציאת הקורדינטות של מיקום המצלמה בעולם.  
אני משתמש ב **Nominatim** geopy.geocoders import כי הוא בחינם.

אשתמש בשפת SQL כדי לבצע פעולות על databasen.  
אממש פרוטוקול TLS משלי בהקמת החיבור בין השרת ללקוח ובהמשך בין העברת המידע.

## 2.1 טכנולוגיות בשימוש בפרויקט

הפרויקט כולל שימוש במאגרי מידע מסוג sql לשמירת הרשומים ולשם שמירת תמונות המבוקשים בצד השרת וגם בצד הלקוח.  
הפרויקט משתמש בפרוטוקול TLS לתקשורת בין השרת והלקוח והצפנה- למניעת התקפת MITM.  
בגלל שהפרויקט מעביר מאגרי מידע הדבר הכרחי למניעת גניבת מידע.  
בנוסף, הפרויקט מגן מהתקפת SQL INJECTION על ידי שימוש ב"prepared statements" שימוש בטכניקה הזאת מאפשרת הפרדה בין קלט המשתמש לשורת הפקודה למאגר הנתונים.  
הפרויקט גם מגן נגד סוג של מתקפת DOS, SYN FLOOD בכך שבעבור כל לקוח יהיה סופר פקטות SYN / ACK שנישלחו על ידיו כך שאם יעבור מספר מסוים של פקטות SYN ללא שליחת כמות מסוימת של פקטות ACK אז הקשר איתו ינותק. בנוסף לכך, בצד הלקוח ההתחברות נעשית עם captcha למניעת התחברות של בוטים למערכת והצפה של השרת.  
לכל לקוח תהיה מצלמה דרכה הוא מקבל את הווידאו לזיהוי.  
לשרת יש שני מסדי נתונים (sqlite):  
1. מאגר של תמונות של מבוקשים והשמות שלהם.  
2. מאגר של פרטי התחברות.  
ללקוח יהיה גם מסד נתונים של תמונות המבוקשים ושמןם.  
הפרויקט משתמש בספרייה Face recognition לשם זיהוי הפנים. הספרייה מציעה כלים ופונקציות העושות את זיהוי הפנים פשוט יותר ומדויק מאשר לבנות עצמאית רשת נוירונים וללמד מכונה לזהות פנים.

## 2.2 מדריך למשתמש

ראשית חובה להיות מותקן קומפיילר של פייתון 3+.  
כל הספריות שצינתי קודם צריכות להיות מתוקנות. בנוסף חובה להוריד visual studio for c++ כי face\_recognition משתמשת ב C++ וגם את ספריית Dlib שספריית זיהוי הפנים בנויה מעליה והיא למעשה ספרייה למachine learning.  
על השרת והלקוח להיות בעלי מצלמות, ועל התיקיות והקבצים שאצרף להיות במבנה כפי שאראה בתרשים הקבצים.

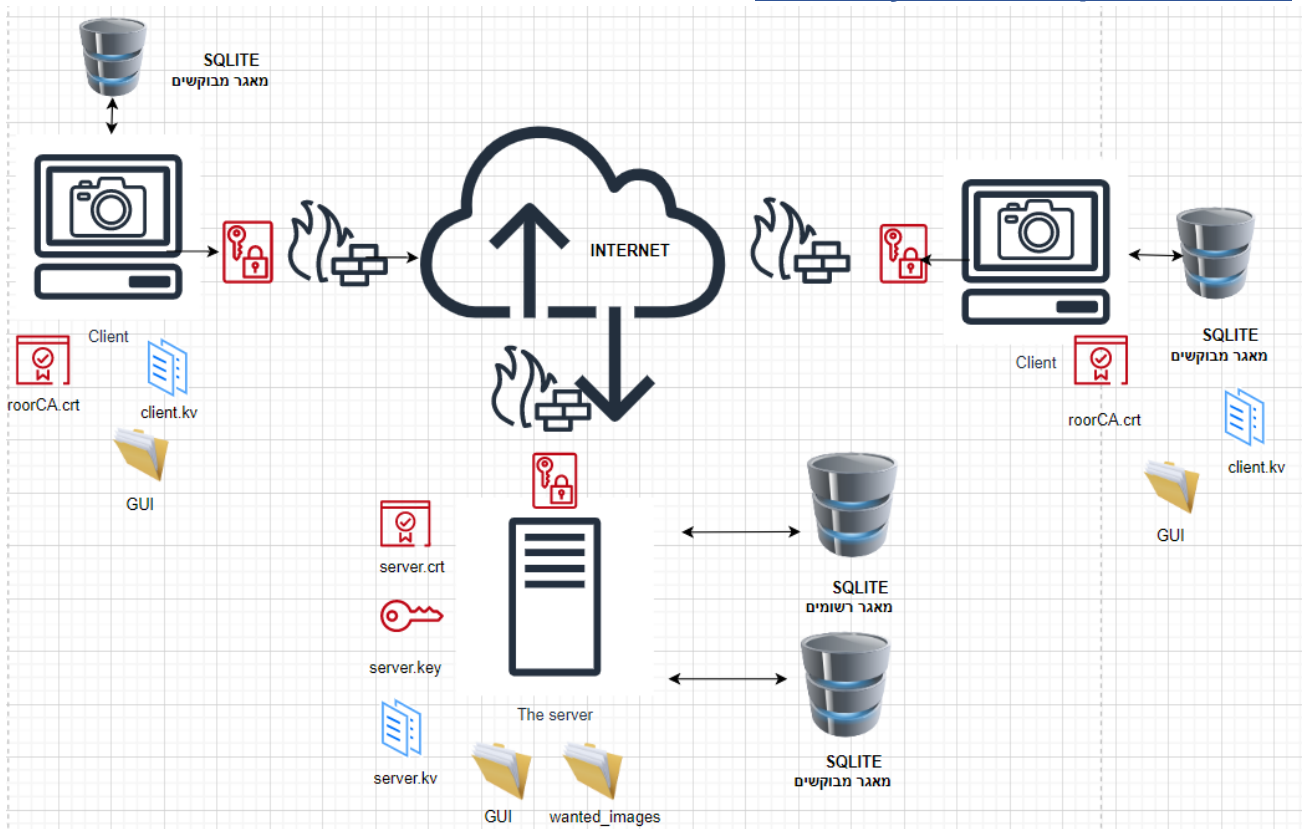


### 3 אפיון דרישות וארכיטקטורת מערכת

#### 3.1 דרישות ושימושי מערכת – Use Cases

על המערכת להתמודד עם כמות רבה של לקוחות (מצלמות).  
 המערכת צריכה להיות מסוגלת לזהות פנים.  
 המערכת צריכה לדעת להתמודד עם תקלות -הלקוח או השרת לעולם לא ייפלו, לדוגמא לקוח נפל אז דבר זה לא אמור להשפיע על השרת או על שאר הלקוחות.  
 על הממשק למשתמש להיות קל ונגיש.  
 על התקשורת בין השרת ללקוח חייבת להיות מוצפנת.

#### 3.2 סביבת הפרויקט – Eco – System



השרת משתמש בפרוטוקול TLS לצורך וידוא אמינות והצפנת המידע. לשרת קיימים שני מאגרי מידע מסוג SQLite:  
 1. login.db - מאגר הרשומים. על מנת להתחבר לשרת עליך להיות בעל מספר זהות יחודי וסיסמא אשר מופיעים במאגר.

2. `wanted_pic_and_name.db` - מאגר המבוקשים. המאגר מחזיק את כל המבוקשים שהוזנו בשרת. הוא מורכב מעמודה של תמונה ועמודה של שם.  
ללקוח גם יש מאגר מידע מסוג `client_db.db` SQLITE הוא זהה למאגר המידע של המבוקשים בשרת. אחרי הקמת הקשר בין השרת ללקוח, השרת שולח ללקוח את מאגר המבוקשים העדכני. החיבור מתבצע כאשר הלקוח מזין את ה `captcha` הנכון ונותן מקום אמיתי של איפה הוא נמצא. (מקום המצלמה בעולם).

### 3.3 ארכיטקטורת המערכת

## פרוטוקול התקשורת בין השרת ללקוח:





PASSWORD - סיסמא מסוג מחרוזת שהמשתמש הכניס.  
 הנתונים בטבלה מוצפנים בשביל הגנה במידה ומישהו יפרוץ למאגר.

**טבלת wanted info מתוך wanteds\_pic and name.db**

name	picture
------	---------

-NAME שם המבוקש מסוג מחרוזת  
 Picture – תמונת פרצוף המבוקש מסוג BLOB (Binary Large Object)

**מסד הנתונים אצל הלקוח**

בדומה לשרת:

**טבלת wanted info מתוך client.db**

name	picture
------	---------

-NAME שם המבוקש מסוג מחרוזת  
 Picture – תמונת פרצוף המבוקש מסוג BLOB (Binary Large Object)

**3.4 טבלת בדיקה עבור ארכיטקטורה ודרישות מערכת**

הדרישה	מימוש הדרישה
על המערכת להתמודד עם כמות רבה של לקוחות(מצלמות).	המערכת פותחת תהליך לכל לקוח וכך ניתן לתקשר עם הרבה לקוחות בו זמנית.
המערכת צריכה להיות מסוגלת לזהות פנים.	המערכת משתמשת בספרייה מיוחדת לכך אשר מבצעת זיהוי פנים.
המערכת צריכה לדעת להתמודד עם תקלות - והלקוח או השרת לעולם לא ייפלו.	שימוש בtry, except
על הממשק למשתמש להיות קל ונגיש.	הממשק למשתמש נעשה בקivy. אצל השרת קיים דף עזרה ובו הסברים על המערכת ועל האפשרויות שהיא מאפשרת. ליד כל שדה יש הסבר של מה צריך להזין לתוכו וליד רוב הכפתורים קיים אייקון שמסמל את הפעולה שהכפתור עושה.
על התקשורת בין השרת ללקוח חייבת להיות מוגנת ומוצפנת.	התקשורת בין השרת ללקוח מתבצעת באמצעות פרוטוקול TLS ושימוש בcertificate.

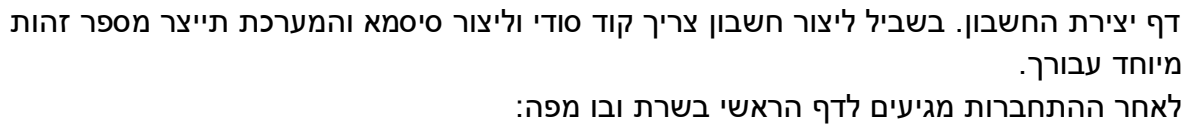
## 4 ממשק משתמש – GUI

הממשק הגרפי בצד השרת וגם בצד הלקוח נעשה בעזרת kivyMD.

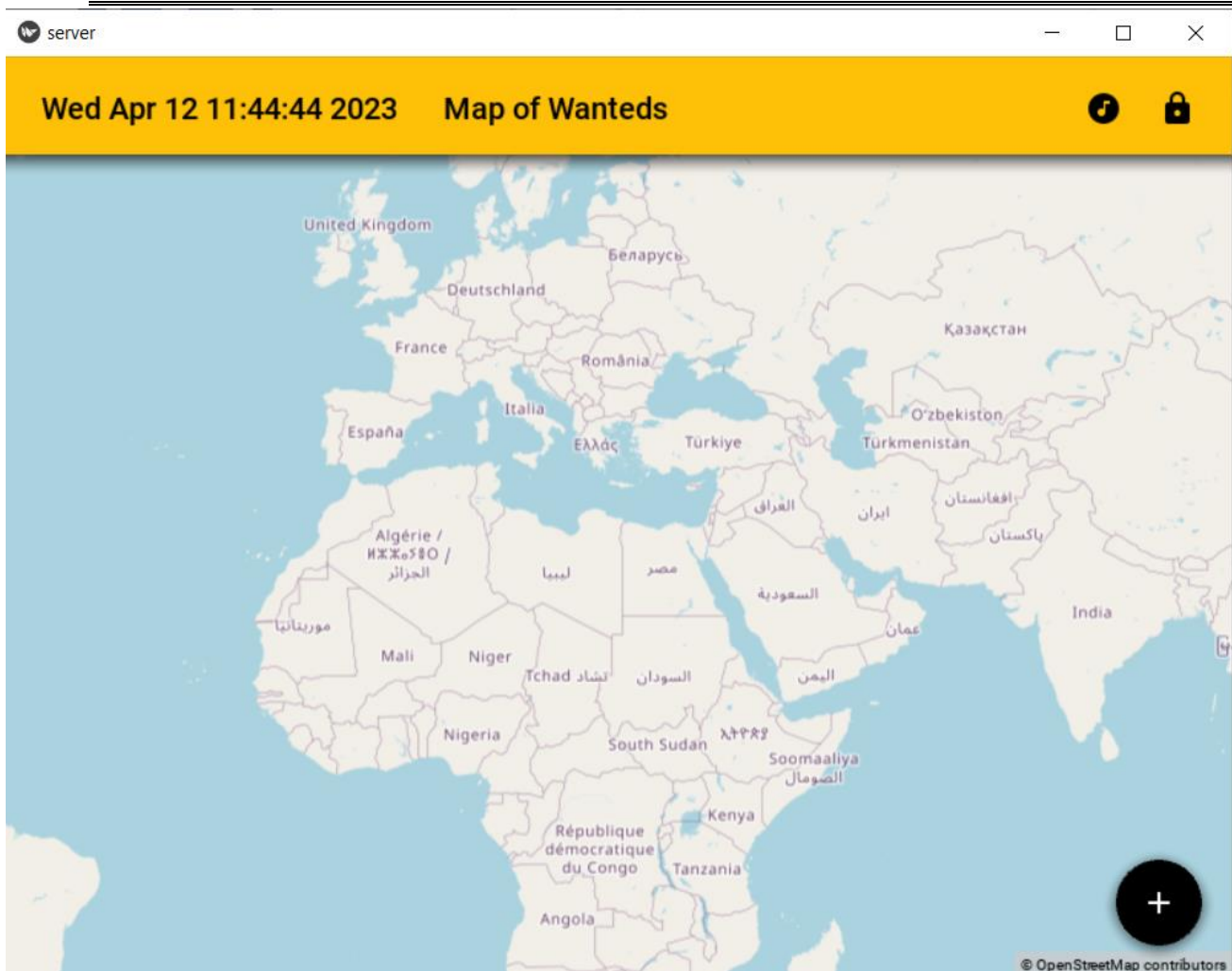
### צד שרת:



דף ההתחברות בשרת. בדף זה ניתן להתחבר אל המערכת וגם ליצור חשבון חדש וכמובן גם לצאת מהמערכת.  
על מנת להתחבר יש להקיש את מספר הזהות וסיסמא.







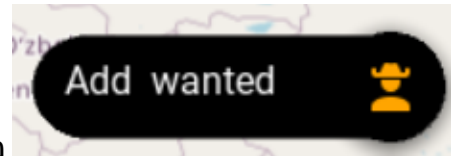
בשורה העליונה יש סימן של מנעול המסמן שהקשר מוצפן וגם יש אפשרות באמצעות לחיצת על לחץ המוזיקה לנגן את השיר "the good the bad and the ugly" בצד ימין למטה יש סימן של + שלחיצה עליו תפתח את סרגל האפשרויות :



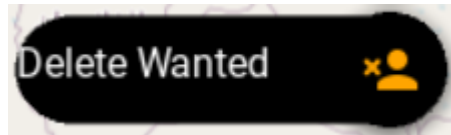
ניתן לשים את העכבר על כל אחד מהאייקונים ויופיע מה הם מאפשרים

לחיצה על כל אחד מהאייקונים תגרום ל:

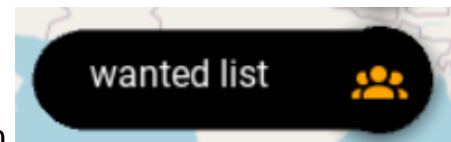




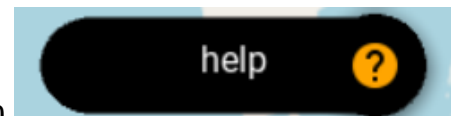
מעבר לדף של הוספת מבוקש



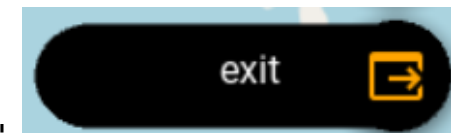
מעבר לדף של מחיקת מבוקש לפי השם שלו



מעבר לדף עם רשימת המבוקשים ותמונה שלהם.



מעבר לדף עם הסבר על המערכת ועל התמצאות בה.



יציאה מהמערכת.

בכל הדפים קיים אייקון של מפה שלחיצה עליו תחזיר לעמוד המפה – הראשי.

### דף הוספת המבוקש:

server

WANTED

FULL NAME:

Enter the wanted's full name here

CRIMES:

Enter the crimes he committed

DANGER LEVEL:

On a scale from 1 to 10 (10 highest)

BOUNTY\$:

Amount of money you offer on his head

PICTURE:

ADD PICTURE FROM GALLERY

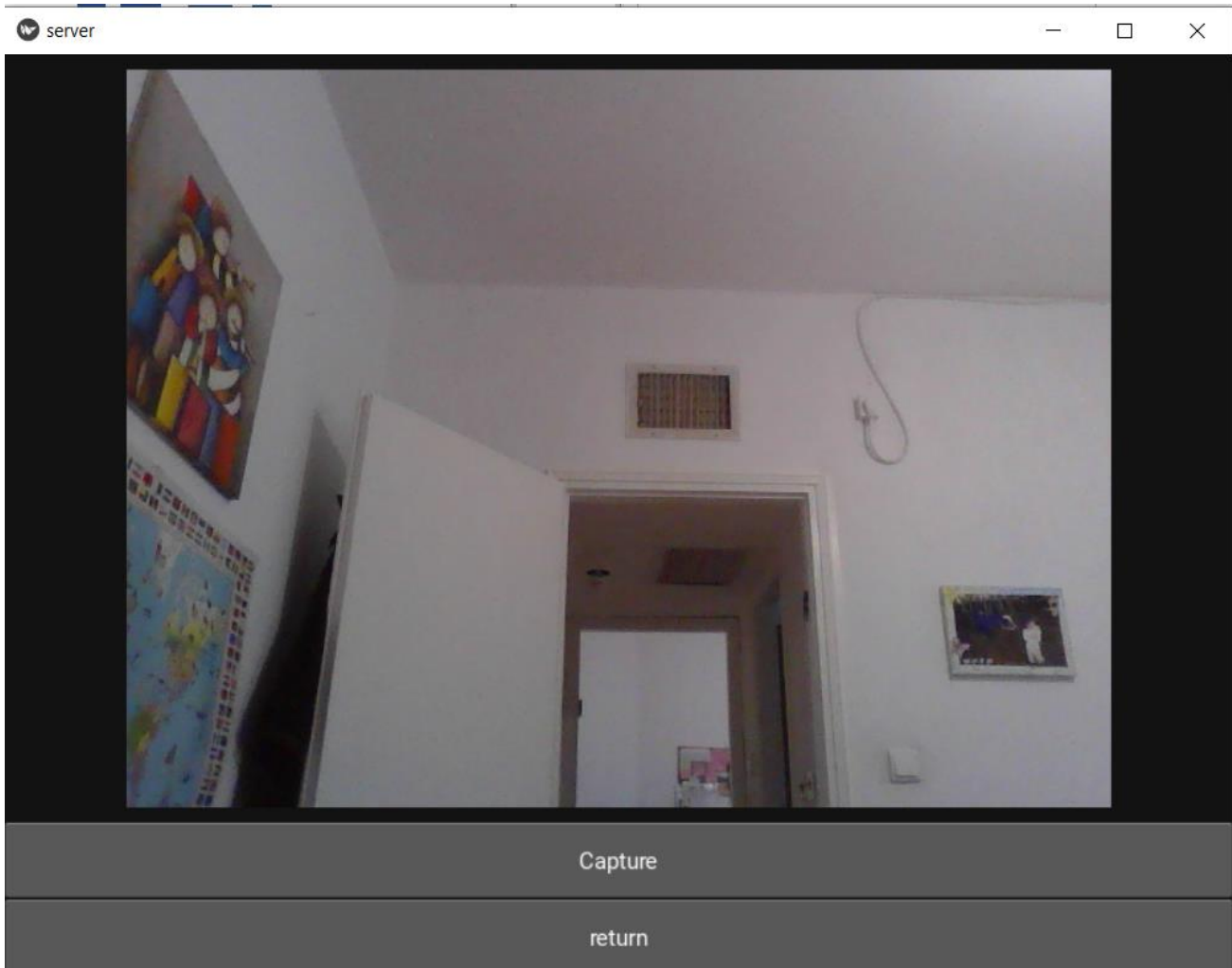
TAKE A PICTURE

SUBMIT

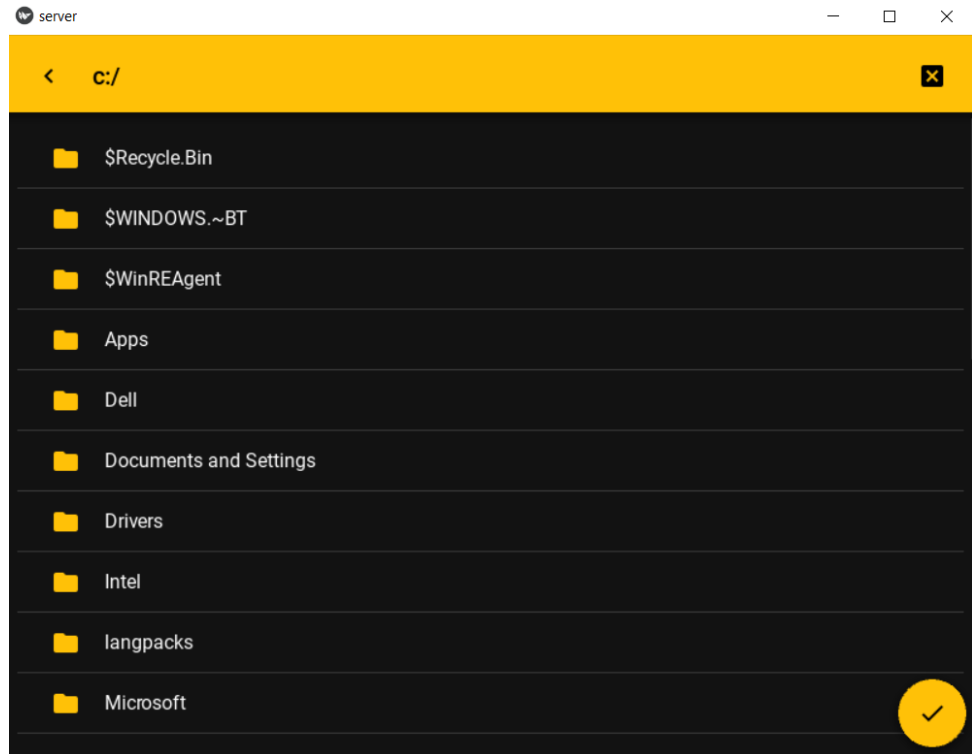
CLEAR

לאחר לחיצה על אייקון "השריף" המערכת תעביר מדף המפה הראשי לדף הנל. הדף מסודר כמו פוסטר של מבוקש כדי שיהיה קל להבנה ומעל כל שדה מופיע בכיתוב קטן מה צריך לרשום בו. בתחתית העמוד מופיע סרגל עם ארבע כפתורים. כפתור ניקוי – מנקה את העמוד כפתור הגשה- שלחיצה עליו תוסיף את המבוקש למאגר המידע והשרת יעדכן את הלקוח להוסיף את המבוקש למאגר שלו. על הנתונים שהוזנו עוברים מספר בדיקות: על השדות לא להיות ריקים על שדה "רמת סכנה" להיות מספר בין 1-10 על שדה הכסף להיות מספר על שדה השם להיות שונה כלומר, אין להכניס מבוקש עם אותו שם פעמיים ולבסוף חייב להוסיף תמונה של פנים של בן אדם ובתמונה חייב להיות רק פרצוף אחד ולא כמה.

כפתור לקיחת תמונה - יעביר לדף של צילום תמונה מהמצלמה שבשרת:

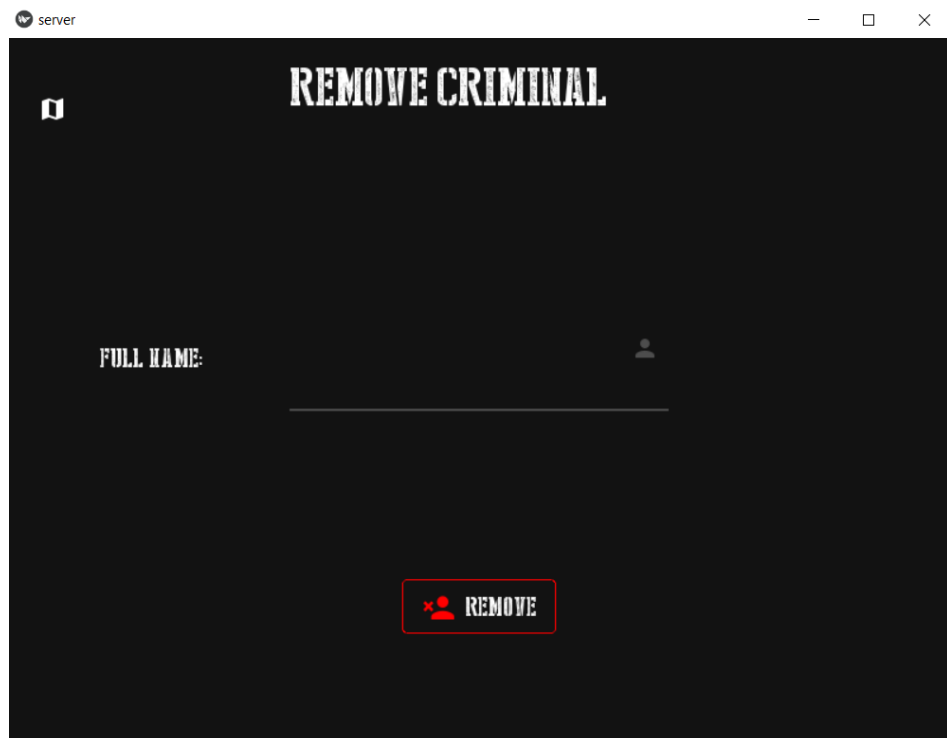


בדף קיימים שני כפתורים - אחד לוקח תמונה והשני בשביל לחזור לעמוד הוספת המבוקש.  
כפתור הוספת תמונה מהגלריה - לחיצה עליו תפתח את הקבצים במחשב בתיקיית c:



ניתן לסייר בין התיקיות ובשביל לבחור קובץ יש ללחוץ עליו.

### דף מחיקת המבוקש:



בדף יש שדה שבו מכניסים את שם המבוקש וכפתור "הסר". אם המבוקש קיים במערכת אזי המערכת תמחק אותו מהמפה (אם מופיע עליה) וממאגר המבוקשים ובנוסף תעדכן את הלקוח שעליו למחוק את המבוקש מהמאגר שלו.

דף רשימת המבוקשים:

בעמוד יוצגו כל המבוקשים באותו הרגע. דוגמא של העמוד כאשר גו ביידן וברק אובמה מבוקשים:

server

— □ ×

דף עזרה:

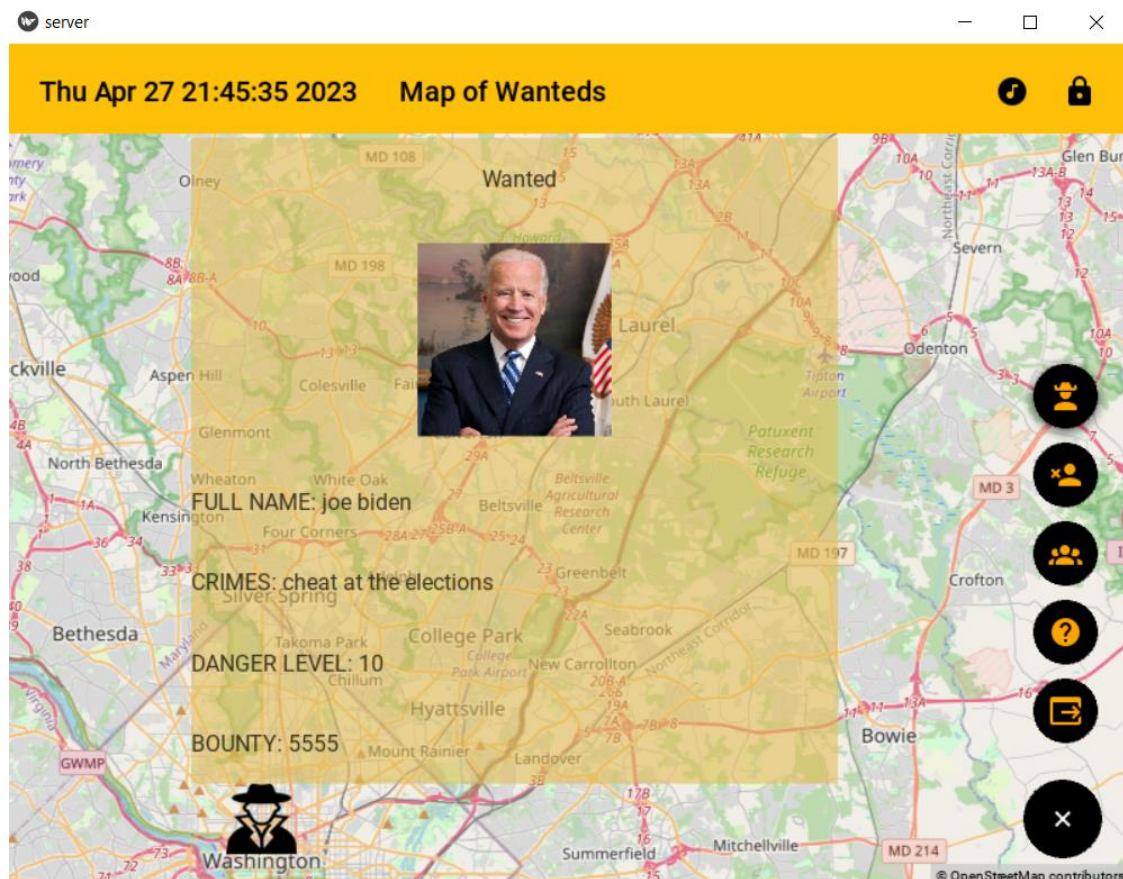
בדף יש הסבר על המערכת ועל כל האפשרויות שהיא מציע.



בנוסף לכל הדפים הללו בדף הראשי של המפה ברגע שלקוח זיהה מבוקש הוא מודיע לשרת והשרת מוסיף אייקון של מבוקש על מפה לפי מיקום המצלמה:



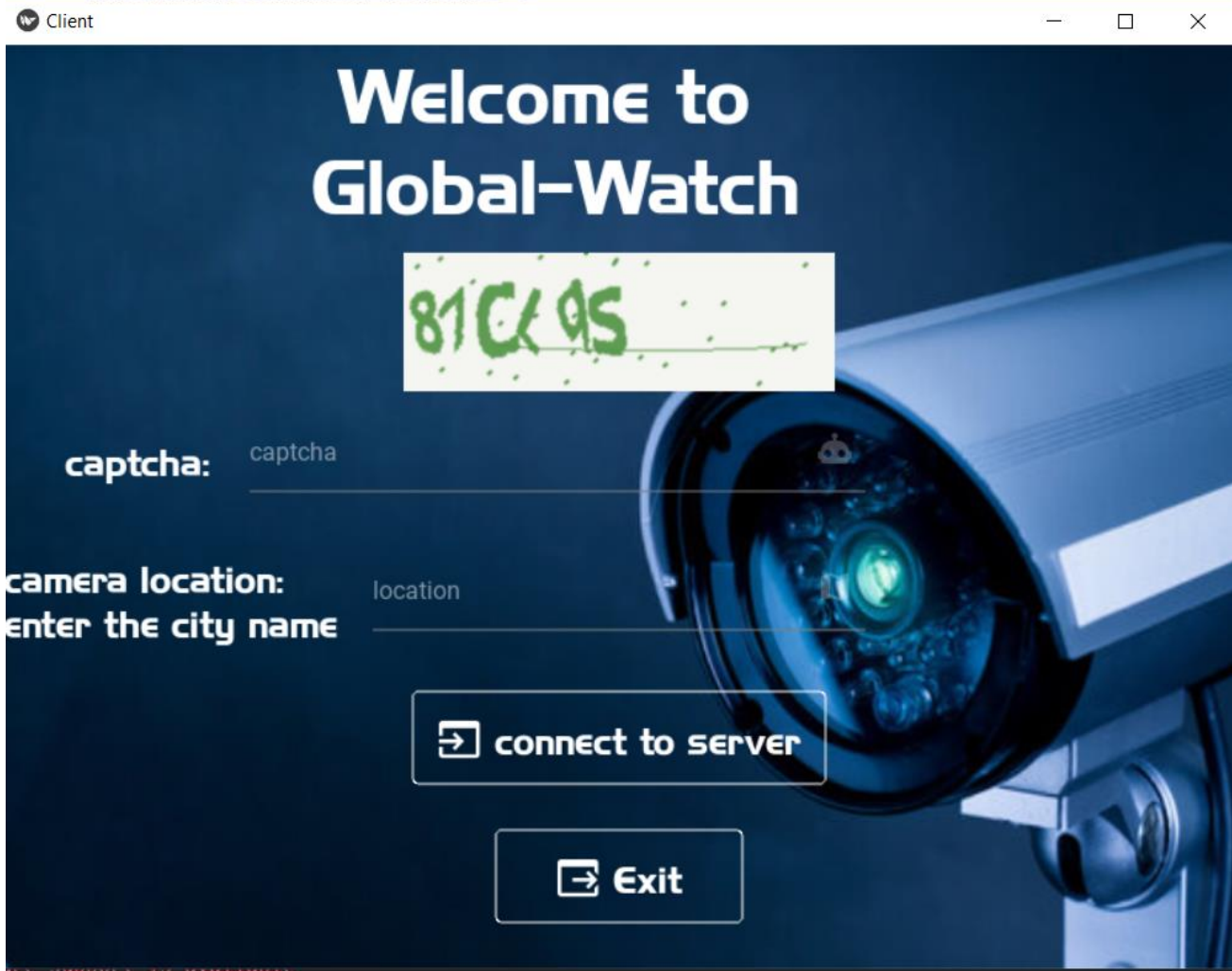
לחיצה על אייקון המבוקש תפתח את המידע עליו:



צד לקוח:



אצל הלקוח אין הרבה גוי ביחס לשרת כי הוא מתפעל כמו מצלמה והגוי הוא רק בשביל הכנסת פרטי המיקום של המצלמה והCAPTCHA דף הלקוח:



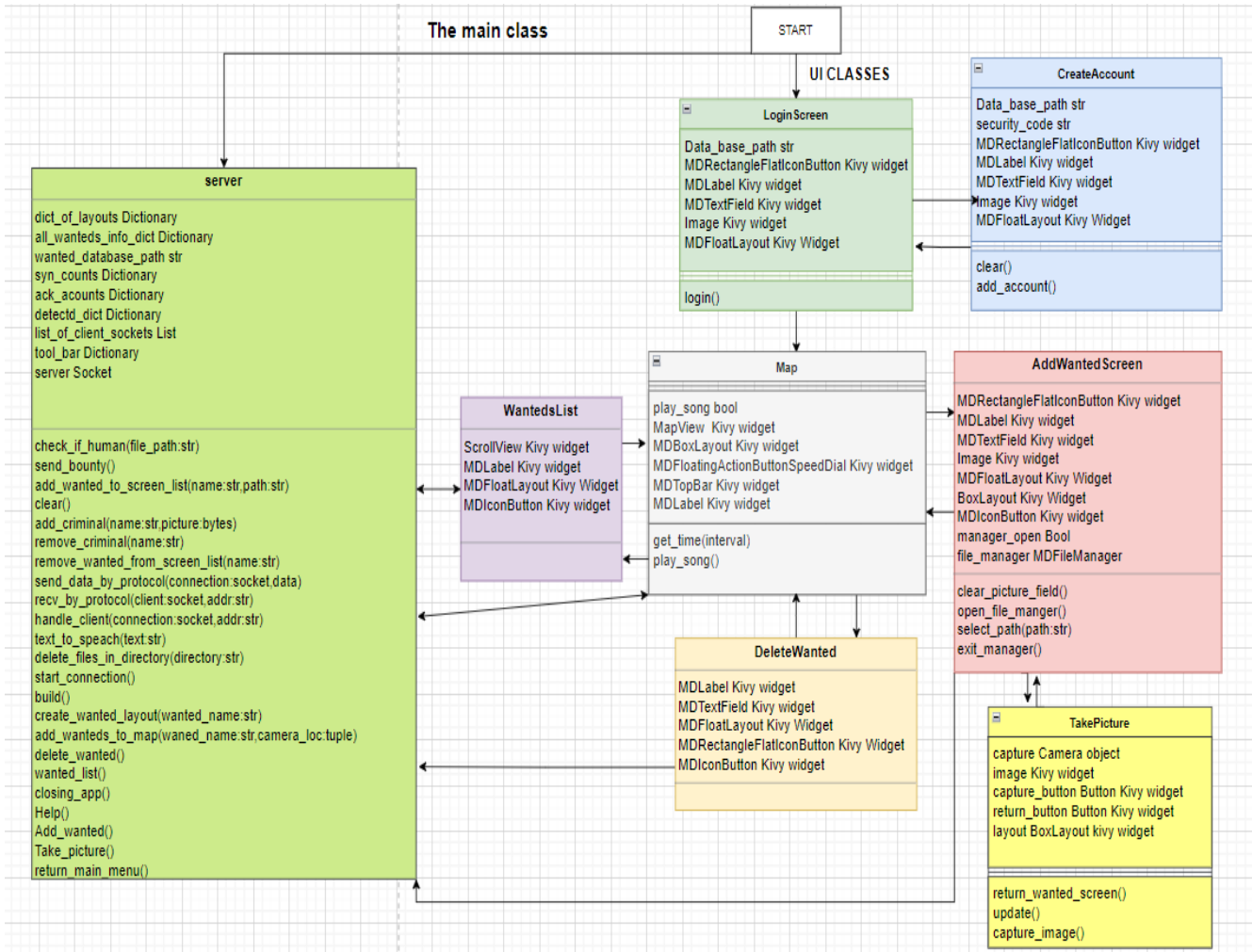
קיימים שני שדות:

1. captcha מעל השדה מופיע התמונה ובה הטקסט אותו יש להעתיק לשדה על מנת לוודא שהלקוח לא רובוט.
  2. שדה המיקום. התוכנה בודקת שהמיקום אכן קיים בעולם ומשיגה את נקודות הציון שלו (latitude, longitude).
- בתחתית העמוד יש שני כפתורים אחד על מנת לתחבר לשרת והשני על מנת לצאת מהדף ולסגור את התוכנית.

## 5 מדריר למפתח

### 5.1 דיאגרמת UML של כל מחלקות הפרויקט והתליות ביניהן

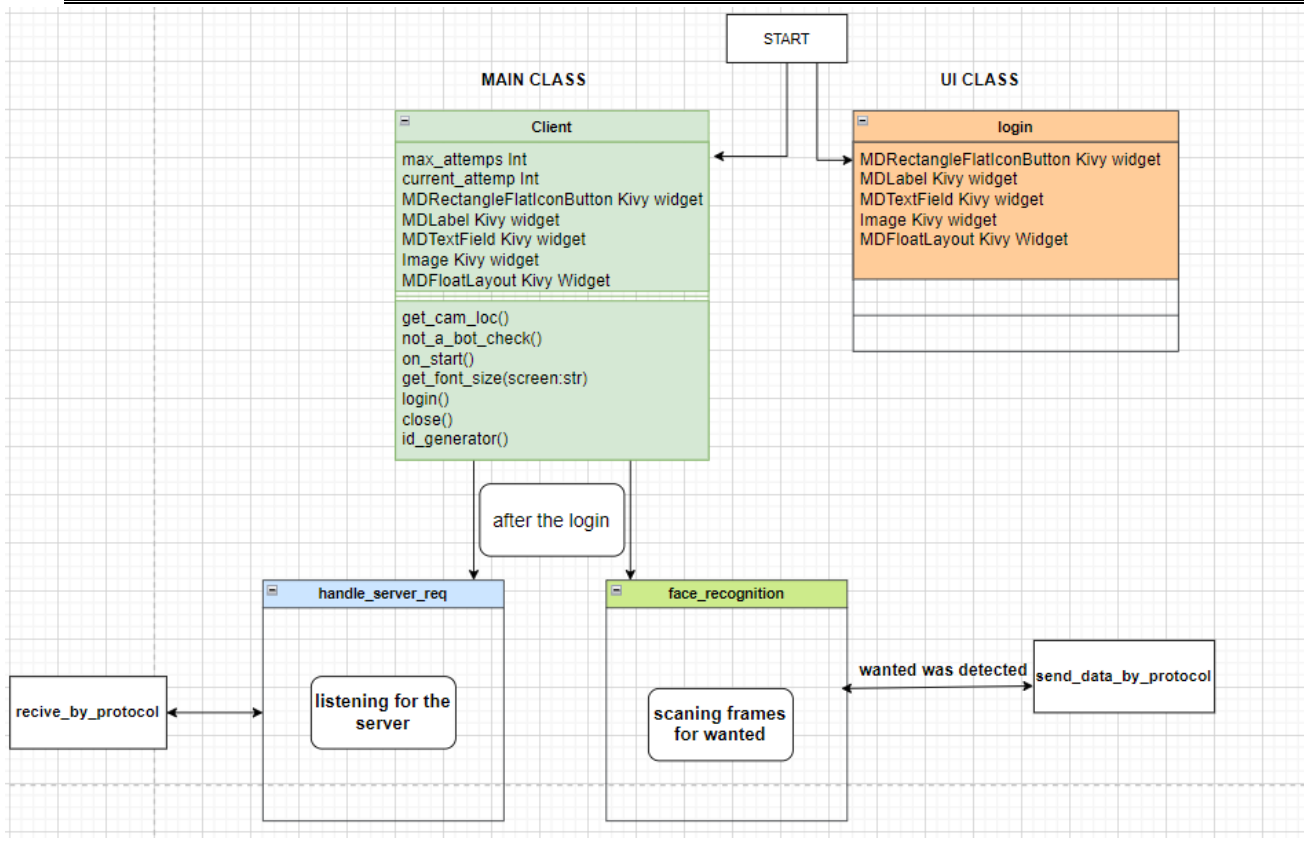
#### בשרת:



אצל השרת רוב המחלקות הן מחלקות העוסקות ב־gRPC ויש רק מחלקה אחת שהיא עוסקת בפונקציונליות עצמה של השרת.

#### לקוח:



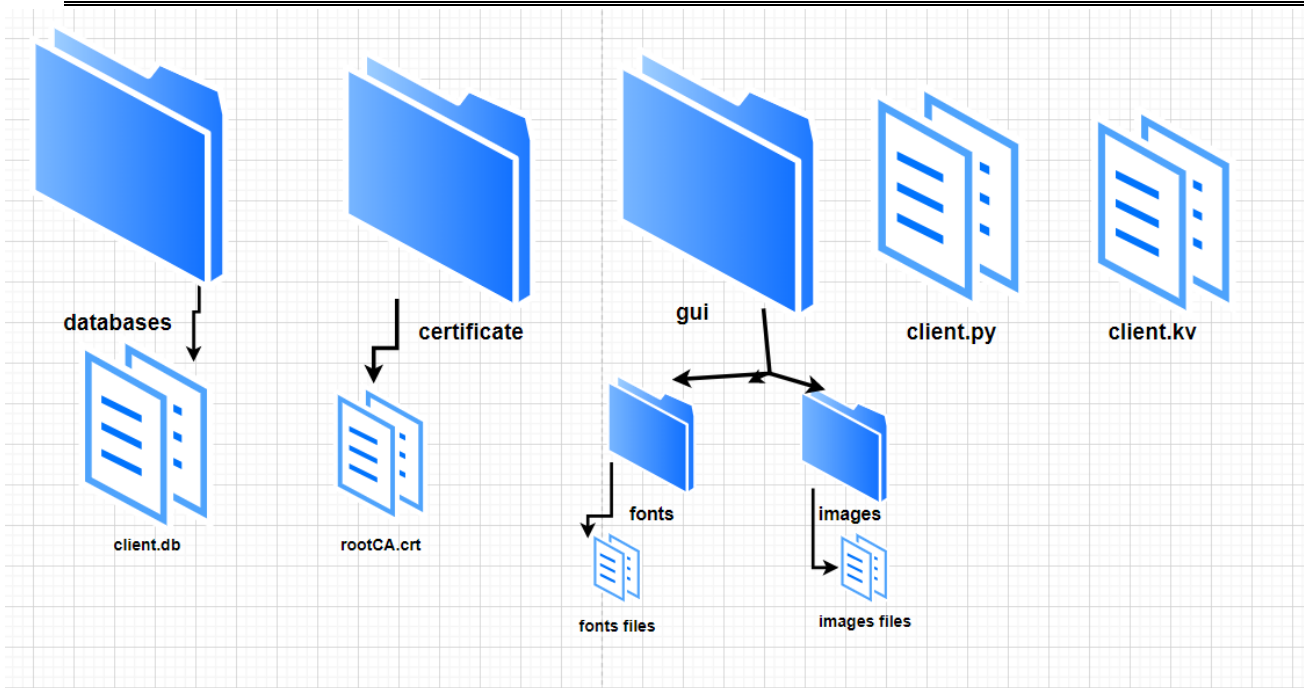


אצל הלקוח יש שתי מחלקות שהן עוסקות בחיבור הלקוח לשרת. שתי המחלקות דומות למחלקת server, LoginScreen אצל השרת. אחרי שעוברים את ההתחברות שני התהליכים מתחילים תהליך של זיהוי פנים ותהליך של הקשבה לבקשות השרת.

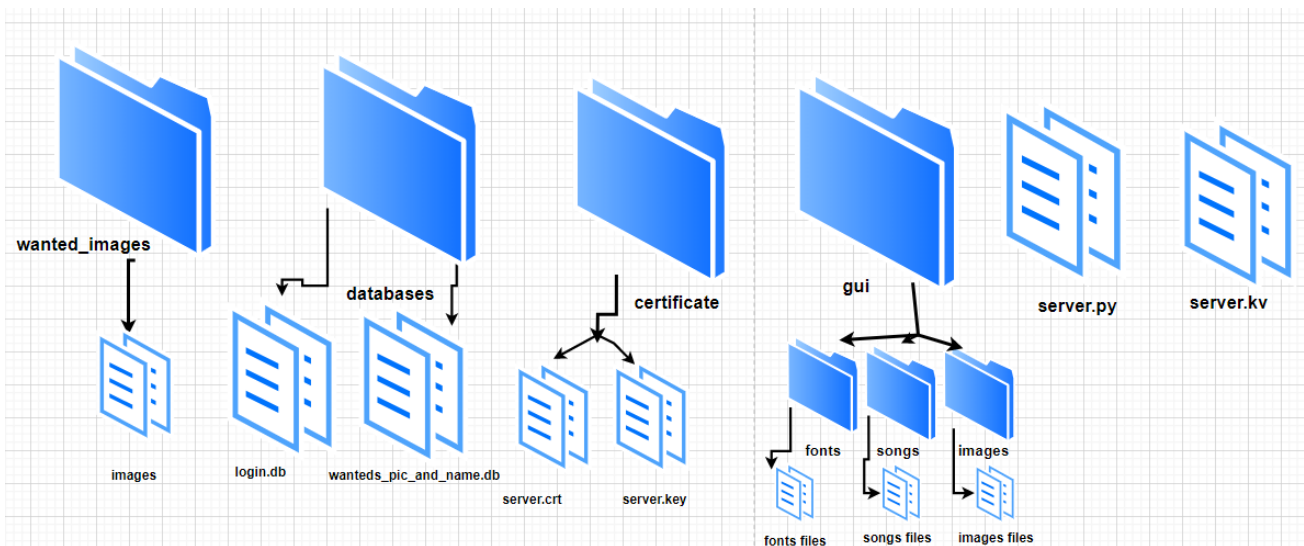
## 5.2 רשימת פונקציות ומחלקות ותפקידיהם

תיאור הקבצים בפרויקט:

לקוח:



שרת:



בתוך תיקיית certificate גם בשרת וגם בלקוח נמצאים הקבצים הדרושים לפרוטוקול TLS.  
בתוך תיקיית databases יש את מסדי הנתונים בשרת יש שנים – אחד לרשומים והשני למבוקשים ואצל הלקוח יש רק אחד והוא למבוקשים.  
בתוך תיקיית wanted\_images מצויים כל התמונות של המבוקשים שהוכנסו למערכת.  
בתוך תיקיית gui נמצא כל הקבצים הגרפיים שאני משתמש בהם בפרויקט.  
בקובץ server.kv יש את הגרפיקה והעיצוב של המחלקות ובקובץ PY פעולות על המחלקה.  
וכך גם בלקוח.

קובץ	מחלקה/פונקציה עיקרית	תפקידה
------	----------------------	--------

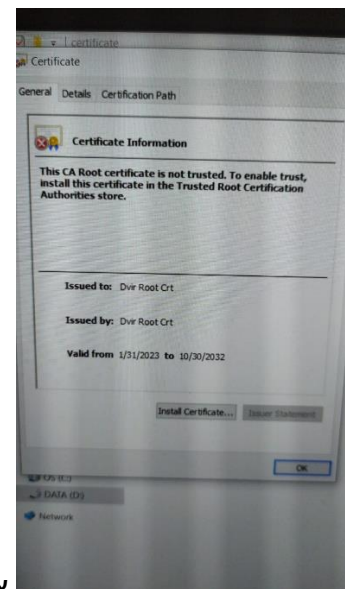
מחלקה של הממשק משתמש בהתחברות. דף ההתחברות.	LoginScreen	server.kv/server.py
מחלקה של הממשק משתמש בדף ההרשמה.	CreateAccount	server.kv/server.py
מחלקה של הממשק משתמש בדף הראשי אצל השרת - דף מפת המבוקשים.	Map	server.kv/server.py
מחלקה של הממשק משתמש בדף הוספת המבוקש.	AddWantedScreen	server.kv/server.py
מחלקה של הממשק משתמש בדף לקיחת התמונה.	TakePicture	server.kv/server.py
מחלקה של הממשק משתמש בדף מחיקת המבוקש.	DeleteWanted	server.kv/server.py
מחלקה של הממשק משתמש בדף רשימת המבוקשים.	WantedList	server.kv/server.py
המחלקה הראשית בשרת שהיא עוסקת בבדיקת תקינות הנתונים שהוזנו במחלקות האחרות ובהתעסקות עם הלקוחות.	Server	Server.py
מחלקה של הממשק משתמש בדף ההתחברות.	login	Client.py/client.kv
המחלקה הראשית בלקוח והיא עוסקת בבדיקת תקינות הנתונים שהוזנו במחלקת login	client	Client.py
פונקציה שעוסקת בזיהוי הפנים. הפונקציה מקבלת את הווידאו מהצלמה וסורקת כל פריים שני בחיפוש אחר מבוקשים ומעדכנת את השרת אם זיהתה מבוקש.	Face_recognition	Client.py
פונקציה שמטרתה היא להקשיב לבקשות השרת ולפעול בהתאם לפיהם.	Handle_server_req	Client.py

## 6

סיכום אישי / רפלקציה

נהנתי מאוד מהעבודה על הפרויקט, היא הייתה מעניינת ואני שמח מהתוצר שיצרתי. עבודה על פרויקט מסדר גודל כזה מחייבת עבודה מסודרת והצבת יעדים ולכן חילקתי לי את הזמן, הצבתי לעצמי משימות קטנות עד שלבסוף הרכיבו את כל הפרויקט. העבודה על פרויקט לימדה אותי לעבוד מסודר, ועל איך ניגשים לפרויקטים גדולים ועל דרכי התמודדות עם קשיים והכי חשוב – לא לוותר. למדתי על איך עובד פרוטוקול TLS על מתקפת INJECTION SQL ועל איך להגן מפניה, הצפנה, מתקפת DDOS, על זיהוי פנים ועל בניית ממשק משתמש בעזרת KIVYMD. במהלך הפרויקט נתקלתי בכמה קשיים. הקושי הראשון שהיה לי הוא להוריד את ספריית face\_recognition. הספרייה face\_recognition היא ספרייה גדולה שבשביל שתעבוד על המחשב צריך להיות מותקן visual studio c++ ועוד ספריות שהיא משתמשת בהם. כשניסיתי להוריד אותה זה עשה לי הרבה בעיות והגעתי למצב שהייתי צריך למחוק את ה interperter ואת pycharm ולהוריד אותם מחדש לפי errors שההורדה הציגה לי. ה interpertern שהיה לי היה bit32 ו pycharm היה 64 ביט וזה עשה בעיות למודל. המסקנה מקושי זה היא שחייבים לעבוד מסודר ולתאם גרסאות.

עוד קושי שהיה לי הוא עם certificate ב TLS. לא היה לי SSL על המחשב והייתי צריך לחפש באינטרנט ולהוריד. וגם אחרי שהצלחתי לייצר certificate היה עליה X והיה כתוב שאי אפשר לסמוך על certificate כמוראה בתמונה:

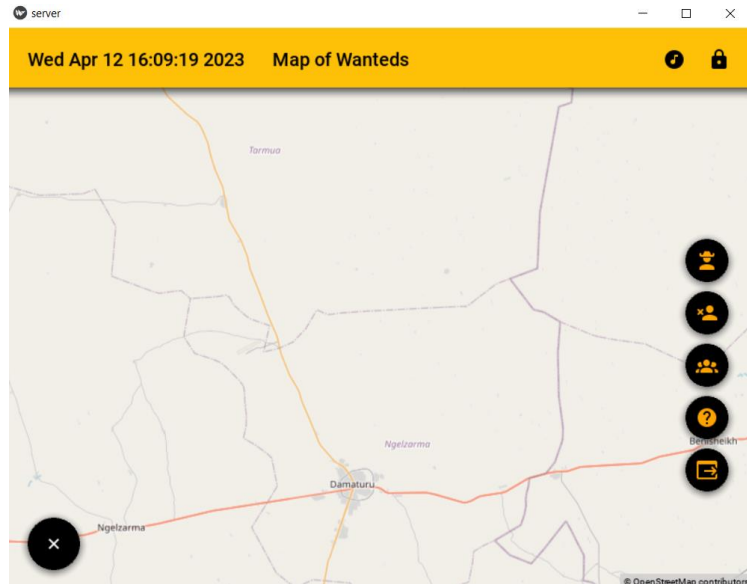


אז כדי לפתור את זה חקרתי ומצאתי סרטון שעזר לי לפתור את הבעיה.

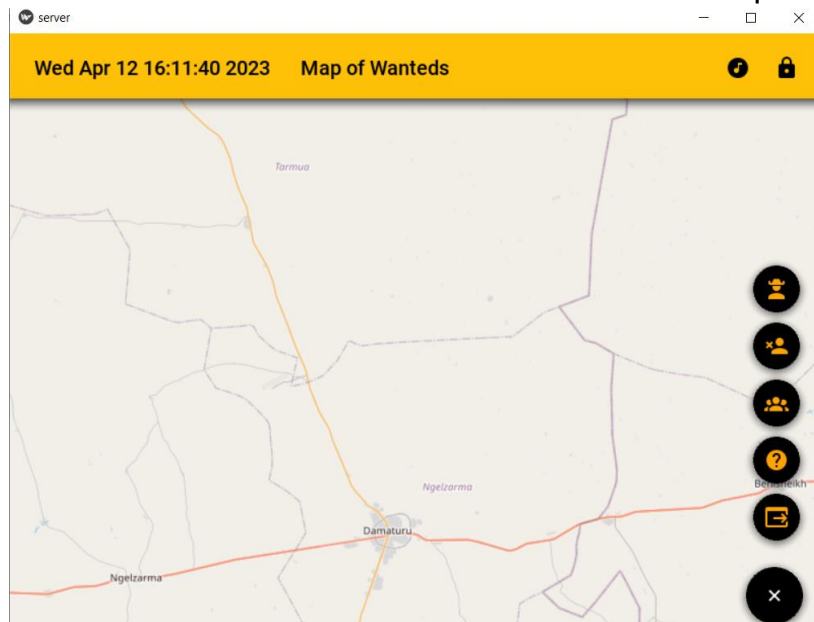
קושי נוסף היה איך לעשות את הזיהוי פנים. קראתי והבנתי מה זה מצריך בשביל לבנות תוכנה שתדע לזהות פנים בדיוק רב. ראיתי סרטונים ביוטיוב וקראתי מאמרים המסבירים על רשת ניורונים סיאמית ועל איך אפשר לבנות מודל לזיהוי בעצמי אך הבנתי שאחוזי ההצלחה של המודל הם נמוכים וגם הוא יהיה איטי (ואני צריך זיהוי פנים מהיר). לאחר מכן, גיליתי את הספרייה face\_recognition שמקלה על התהליך של זיהוי הפנים וויתרתי על הרעיון של לבנות מודל בעצמי. המסקנה מהקושי הזה הוא שכדאי להעזר בספריות מוכנות ושאי לא צריך "לבנות הכול מאפס" כי יש אנשים שעשו זאת כבר בשבילי ואני יכול להשתמש במה שעשו.

בעיה נוספת שהייתה לי היא עם הממשק משתמש. במסך המפה יש סרגל של אפשרויות מסוג

MDFloatingActionButtonSpeedDial. סרגל האפשרויות אמור להפתח בצד שמאל אבל משום מה הוא תמיד היה נפתח אצלי בצד ימין. תמונה של הבאג:



בדקתי וחקרתי בגוגל וחיפשתי במסמך המסודר של kivy ולא היה על זה שום הסבר. לכן התחלתי לנסות לשנות דברים בקוד שלי עד שאני אבין איך לפתור את זה. בסוף גיליתי שאם משנים את גודל המסך זה מחזיר אותו להיות בצד שמאל:



אז תיקנתי את הבאג בעזרת שלוש השורות הבאות:

```
size = Window.size
גודל רנדומלי(350, 650)
Window.size = size
```

אילו הייתי מתחיל את הפרויקט היום הייתי עובד עם הספרייה Dlib שהיא ספריית הבסיס לזיהוי פנים והיא מאפשרת זיהוי פנים יותר מהיר מאשר מה שמציע הספרייה Face\_recognition. בנוסף הייתי מנסה לשדרג את הפרויקט ולגרום לו לעבוד על מצלמות אבטחה אמיתיות.

## 7 מקורות מידע / ביבליוגרפיה

Face recognition python:

<https://www.analyticsvidhya.com/blog/2021/11/build-face-recognition-attendance-system-using-python>

general information on face recognition:

<https://grekkom.com/en/products/our-analytics/facial-recognition/access-control>

kivymd documentation - <https://kivymd.readthedocs.io/en/1.1.1>

face recognition documentation - <https://face-recognition.readthedocs.io/en/latest>

certificaten על X שיש - <https://youtu.be/CqkPraRDuHY>  
- [https://public.cloud.myinfo.gov.sg/docs/OpenSSL\\_installation\\_guide.pdf](https://public.cloud.myinfo.gov.sg/docs/OpenSSL_installation_guide.pdf)

בשביל להוריד SSL

github - <https://github.com>

stackoverflow- <https://stackoverflow.com>

kivy mapview - <https://kivy-garden.github.io/mapview/mapview.html>

pypi for donlowads python modules - <https://pypi.org>

open cv documentation [https://docs.opencv.org/4.x/d6/d00/tutorial\\_py\\_root.html](https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html)  
geopy documentation for (lat,lon) <https://geopy.readthedocs.io/en/stable>

**נספח:****קטעי קוד נבחרים:****הפונקציה הראשונה היא הפונקציה אצל הלקוח שבה מתבצע זיהוי הפנים:**

```
def face_recogintion(client):
    """reciving the client socket and detecet all of the wanted faces that
    will appear on the camera video. if a
    wanted is detected sending a msg to the server """
    while True:

        if len(known_face_encodings) > 0 and len(known_face_names) > 0:

            face_locations = []
            Detected_names = []
            process_this_frame = True
            # video_capture = cv2.VideoCapture(1)
            while True:
                try:
                    scale = 10
                    # Grab a single frame of video

                    ret, frame = video_capture.read()
                    if not ret:
                        break
                    height, width, channels = frame.shape
                    # the zoom part:
                    centerX, centerY = int(height / 2), int(width / 2) #
coordinates of the center of the frame.
                    radiusX, radiusY = int(scale * height / 100), int(
                        scale * width / 100) # the desired scale of the zoom
- how many rows and columns to take from the center of the frame
                    # the boundaries of the cropped image based on the center
and radius:
                    minX, maxX = centerX - radiusX, centerX + radiusX
                    minY, maxY = centerY - radiusY, centerY + radiusY
                    # the original frame within the boundaries - (minX:maxX,
minY:maxY)
                    # i tell it from what x to what x the new frame will be
and the same for the y:
                    cropped = frame[minX:maxX, minY:maxY] # crop the image -
shrink it
                    frame = cv2.resize(cropped, (width,
                                                height)) # here the zoom
affect happen - make the crop img on the orignile frame size - we stretch the
image.
                    frame = cv2.flip(frame, 1) # mirorr the frame not
neccesary for face recognition
                    # Only process every other frame of video to save time
                    if process_this_frame:
                        # resize frame of video to 1/4 size for faster face
```

```

recognition processing
    # the smaller the frame is the faster face
recognition processing will be.
    small_frame = cv2.resize(frame, (0, 0), fx=0.25,
fy=0.25)

    # convert the image from BGR color (which OpenCV
uses) to RGB color (which face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]#the code
uses NumPy array slicing. The[:, :, ::-1] slice operation is applied to
reverse the order of the color channels, swapping the blue and red channels.

    # Find all the faces in the current frame of video
    face_locations = face_recognition.face_locations(
        rgb_small_frame) # Returns a 2d array of
bounding boxes of human faces in a image using the CNN face detector
    # cnn detector - CNN face detector works by taking an
input image and running it through a deep
    # neural network specifically designed for detecting
faces. The network is trained on a large
    # dataset of labeled face images to learn patterns
and features that are indicative of a face.
    # The network outputs a set of bounding boxes that
indicate the location and size of the detected
    # faces in the input image.
    face_encodings =
face_recognition.face_encodings(rgb_small_frame,

face_locations) # return a list of 128-dimension face encoding for each face
in the image.

    Detected_names = []

    if not (len(known_face_encodings) > 0 and
len(known_face_names) > 0):

        break
    with lock:
        for face_encoding in face_encodings:
            # See if the face is a match for the known
face(s)

            matches =
face_recognition.compare_faces(known_face_encodings,
face_encoding,

tolerance=0.6) # Compare a list of face encodings against a candidate
encoding to see if they match.

            # returns a list of true and false - true
there is a face that match, false no match
            if True in matches: # if a match was found

```



```

between the current face and the other faces:
    # now what we will do is find the closest
match the face that is the most similar

    # Given a list of face encodings, compare
them to a known face encoding and get a
    # euclidean distance(the distance between
two points) for each comparison face. The
    # distance tells you how similar the
faces are.return a numpy ndarray(it is similar
    # to a list or a Python array, but unlike
them, NumPy ndarrays can hold data of
    # different types, including numbers,
characters, and even other Python objects.)
    # with the distance for each face in the
same order as the 'known_face_encodings' list:
    face_distances =
face_recognition.face_distance(known_face_encodings, face_encoding)

    best_match_index = np.argmin(
        face_distances) # return the index
of the min distace so the similarity are the closes.
    # It is used to find the index of the
minimum element in a NumPy array

    # so best_match_index is give me the most
similaar face but it might be that the the minunimu - the closet face is not
a match at all is not similar at all thats why i checked if there are True in
matches

    if matches[best_match_index]:

        name =
known_face_names[best_match_index] # get the name
        Detected_names.append(name)

    process_this_frame = not process_this_frame

    # Display the results
    for (top, right, bottom, left), name in
zip(face_locations, Detected_names):
        # Scale back up face locations since the frame we
detected in was scaled to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom),
(0, 0, 0), 2)

        # (image, start_point, end_point, color, thickness)

```

```

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom - 35), (right,
bottom), (0, 0, 0), cv2.FILLED)

        cv2.putText(frame, name, (left + 6, bottom - 6),
cv2.FONT_HERSHEY_DUPLEX, 0.8, (0, 0, 255), 1)
        #(image, text, the coordinates of the bottom-left
corner of the text string in the image, font, fontScale, color[, thickness[,
lineType[, bottomLeftOrigin]])
        send_data_by_protocol(client, ["DETECTED", name])

    # Display the resulting image
    cv2.imshow('face recognition video', frame)

    cv2.waitKey(1) # adding a delay of a 0.001 seconds
except ConnectionRefusedError:
    print("closing connection- the server is closed")
    cv2.destroyAllWindows()
    client.close()
    os._exit(0)
    return
except:
    print("closing connection")
    cv2.destroyAllWindows()
    client.close()
    os._exit(0)

else:
    cv2.destroyAllWindows()
    time.sleep(2)

```

### הפונקציה הבאה היא פונקצית קבלת ההודעות אצל השרת ובדיקת SYN FLOOD

```

def send_data_by_protocol(self, connection, data_to_transfer):
    """recive the sockect and the data.
    the protocol of communication between the server and the client"""
    HEADERSIZE = 10

    data = pickle.dumps(data_to_transfer)
    # the msg according to the protocol - the length of the data is in the
header and the maximum len is number
    # with 10 digits and after the header comes the data itself
    msg = bytes(f"{len(data):<{HEADERSIZE}}", 'utf-8') + data
    connection.send(msg)

def recv_by_protocol(self, client, addr):
    """recive the client socket and his adres and receiving msg by the

```

```

protocol while defending against syn
flood """
HEADERSIZE = 10
full_msg = b''
new_msg = True
while True:
    msg = client.recv(4096)
    if not msg:
        break

    if msg.startswith(b"\x16\x03"):
        # If this is a TLS handshake message, reset SYN and ACK counts
        for this client
            self.syn_counts[addr] = 0
            self.ack_counts[addr] = 0

    if msg[13] & 0x02: # Check if packet is a SYN packet (SYN flag is
set in 14th byte of TCP header)
        # If this is a SYN packet, increment SYN count for this client
        self.syn_counts[addr] = self.syn_counts.get(addr, 0) + 1
        print(f"SYN packet received from {addr} and this is count
{self.syn_counts[addr]}")

    if msg[13] & 0x10: # Check if packet is an ACK packet (ACK flag is
set in 14th byte of TCP header)
        # If this is an ACK packet, increment ACK count for this client
        self.ack_counts[addr] = self.ack_counts.get(addr, 0) + 1
        print(f"ACK packet received from {addr} and this is count
{self.ack_counts[addr]}")

    if self.syn_counts.get(addr, 0) > 10 and
self.ack_counts.get(addr, 0) < 2:
        # If SYN flood is detected for this client, terminate
connection
        print(f"SYN flood detected from {addr}. Terminating
connection.")
        # Remove SYN and ACK counts for this client
        del self.syn_counts[addr]
        del self.ack_counts[addr]
        self.list_of_client_sockets.remove(client)
        # Close the connection
        client.shutdown(socket.SHUT_RDWR)
        client.close()
        print(f"Connection with {addr} closed")
        return "CLOSE"

    if new_msg:
        print("new msg len:", msg[:HEADERSIZE])
        msglen = int(msg[:HEADERSIZE])
        new_msg = False

```

```

print(f"full message length: {msglen}")

full_msg += msg

print(len(full_msg))

if len(full_msg) - HEADERSIZE == msglen:
    print("full msg recvd")
    finale_msg = pickle.loads(full_msg[HEADERSIZE:])
    break
return finale_msg

```

### הקטע קוד הבא יוצר את מאגר הנתונים בלקוח ואת הרשימות לזיהוי הפנים

```

def create_database():
    """create the images database of all the wanteds"""
    try:
        with open('database/client_db.db', 'wb') as f:
            f.write(database_of_wanted)

        conn = sqlite3.connect("database/client_db.db")
        cursor = conn.cursor()
        # execute a query to select data from the database
        cursor.execute('SELECT * FROM wanted_info')
        cursor.row_factory = sqlite3.Row
        # iterate over the rows of the result set
        for row in cursor:
            # convert row to a list
            row_list = list(row)
            img_numpy_arr = cv2.imdecode(np.frombuffer(row_list[1],
np.uint8), 1)
            face_encoding = face_recognition.face_encodings(img_numpy_arr)[0]
            known_face_encodings.append(face_encoding)
            known_face_names.append(row_list[0])
            print(f"Recived the database with - {len(known_face_names)} wanteds")
            print("face recognition lists are set!")
            conn.close()
    except Exception as error:
        print(error)
        cv2.destroyAllWindows()
        client.close()
        os._exit(0)

```

### הפונקציה שבדקת את קלט והלקוח לגבי מיקומו בעולם:

```

def get_cam_loc(self):
    """reciving the camera location from the ui """
    try:
        global camera_loc
        # Initialize Nominatim API
        geolocator = Nominatim(user_agent="Global_Watch")

```

```
cam_location_name = self.root.ids.login.ids.loc.text
cam_location = geocator.geocode(cam_location_name)
if cam_location:
    # entering the location name

    print(cam_location.address)

    # printing latitude and longitude
    print(f"Latitude = {cam_location.latitude} ")
    print(f"Longitude = {cam_location.longitude}")
    camera_loc = [cam_location.latitude, cam_location.longitude]
    return True

else:
    self.root.ids.login.ids.loc.helper_text = "Enter a real place"
    self.root.ids.login.ids.loc.error = True
    print("Enter a real place")
except Exception as error:
    print(error)
    cv2.destroyAllWindows()
    client.close()
    os._exit(0)
```