הטכניון – מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל

Laboratory of Networked Software Systems

**בית חכם עם רשת חברתית**

אביב תשע"ח

מגיש: אלכסיי חרומוב מנחה: אורן קלינצקי

תוכן עניינים

[1. מבוא 3](#_Toc528696846)

[2. אתגרים בפיתוח אפליקצייה חכמה 4](#_Toc528696847)

[3. SDKs & APIs 5](#_Toc528696848)

[facebook-jssdk 5](#_Toc528696849)

[AWS - Rekognition 5](#_Toc528696850)

[BEAUTIFULSOUP 5](#_Toc528696851)

[FLASK... 6](#_Toc528696852)

[OPENCV4 + NUMPY 6](#_Toc528696853)

[4. מבט על... 7](#_Toc528696854)

[5. ארכיטקטורה ומימוש 9](#_Toc528696855)

[6. סיכום 17](#_Toc528696856)

[7. ביבליאוגרפיה... 17](#_Toc528696857)

[8. נספחים 17](#_Toc528696858)

1. מבוא

הטכנולוגיה נמצאת בבית של כל אחד ואחת מאיתנו, כולנו נושאים אותה בכיס, ואפילו עונדים אותה במקום שעון קוורץ. אז למה לא להכריח אותה להציג לנו מראש את מה שאנחנו אוהבים?

ניקח דוגמה: אדם קם בבוקר, רוצה לראות מה חדש בעולם החדשות, הספורט או הסדרות האהובות עליו, הוא כבר עשה "לייק" לעמודים האהובים עליו בפייסבוק, נשאר רק שהמחשב יציג בשבילו את העדכונים החדשים מתוכם!

כאן הבית החכם נכנס לפעולה: נעמיד מחשב 3raspberry pi עם מצלמה ומסך מחוברים אליה, המצלמה תקלוט ותזהה את האדם העומד מולה, המסך יציג את התוכן האהוב עליו מאחד עמודי פייסבוק.

כעת נניח שהאדם קרא את העדכון הכי חדיש, והוא רוצה לעבור לפוסט הבא, או שהוא דווקא רוצה להחליף עמוד, למה "להתאמץ" בלחיצת כפתורי מקלדת \ עכבר, אם אפשר פשוט לעשות תנועות ידיים באוויר?

נכון! המערכת תזהה את תנועות הידיים ותפעל לפיהם.

1. אתגרים בפיתוח אפליקצייה חכמה

בפיתוח האפליקציה עמדו לפנינו אתגרים רבים. ראשית התממשקות עם SDKs שונים (יפורטו בהמשך), עמידה בדרישות אבטחה של פייסבוק, זיהוי תנועות ידיים בעזרת מצלמה ולבסוף שימוש במחשב raspberry pi3 לתפעול האפליקציה.

* האתגר הכי גדול בפיתוח התגלה דווקא לאחר ניסוח הפרוייקט:

מתוך ויקיפדיה:

**קיימברידג' אנליטיקה** (ב[אנגלית](https://he.wikipedia.org/wiki/%D7%90%D7%A0%D7%92%D7%9C%D7%99%D7%AA" \o "אנגלית): **Cambridge Analytica**) הייתה [חברה פרטית](https://he.wikipedia.org/wiki/%D7%97%D7%91%D7%A8%D7%94_%D7%A4%D7%A8%D7%98%D7%99%D7%AA) [בריטית](https://he.wikipedia.org/wiki/%D7%91%D7%A8%D7%99%D7%98%D7%A0%D7%99%D7%94) ששילבה [כריית נתונים](https://he.wikipedia.org/wiki/%D7%9B%D7%A8%D7%99%D7%99%D7%AA_%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D), מסחר בנתונים, [ניתוח מידע](https://he.wikipedia.org/wiki/%D7%A0%D7%99%D7%AA%D7%95%D7%97_%D7%9E%D7%99%D7%93%D7%A2) ואסטרטגיה תקשורתית לצורך השפעה על תהליכי [בחירות](https://he.wikipedia.org/wiki/%D7%91%D7%97%D7%99%D7%A8%D7%95%D7%AA) במדינות ברחבי העולם[...] בשנת [2016](https://he.wikipedia.org/wiki/2016) הייתה החברה מעורבת במסע [הבחירות לנשיאות ארצות הברית](https://he.wikipedia.org/wiki/%D7%94%D7%91%D7%97%D7%99%D7%A8%D7%95%D7%AA_%D7%9C%D7%A0%D7%A9%D7%99%D7%90%D7%95%D7%AA_%D7%90%D7%A8%D7%A6%D7%95%D7%AA_%D7%94%D7%91%D7%A8%D7%99%D7%AA_2016) של [דונלד טראמפ](https://he.wikipedia.org/wiki/%D7%93%D7%95%D7%A0%D7%9C%D7%93_%D7%98%D7%A8%D7%90%D7%9E%D7%A4) ובקמפיין ה[ברקזיט](https://he.wikipedia.org/wiki/%D7%91%D7%A8%D7%A7%D7%96%D7%99%D7%98) ליציאת [בריטניה](https://he.wikipedia.org/wiki/%D7%91%D7%A8%D7%99%D7%98%D7%A0%D7%99%D7%94) מ[האיחוד האירופי](https://he.wikipedia.org/wiki/%D7%94%D7%90%D7%99%D7%97%D7%95%D7%93_%D7%94%D7%90%D7%99%D7%A8%D7%95%D7%A4%D7%99" \o "האיחוד האירופי). תפקודה של קיימברידג' אנליטיקה בקמפיינים הללו הוא נושא לחקירות פליליות מתמשכות בשתי המדינות, בעיקר בנוגע לשיטות בהן היא נוקטת לצורך מיקוד בוחרים.]..] ב-[17 במרץ](https://he.wikipedia.org/wiki/17_%D7%91%D7%9E%D7%A8%D7%A5) [2018](https://he.wikipedia.org/wiki/2018), דיווחו העיתונים "[ניו יורק טיימס](https://he.wikipedia.org/wiki/%D7%A0%D7%99%D7%95_%D7%99%D7%95%D7%A8%D7%A7_%D7%98%D7%99%D7%99%D7%9E%D7%A1)" ו"[האובזרבר](https://he.wikipedia.org/wiki/%D7%94%D7%90%D7%95%D7%91%D7%96%D7%A8%D7%91%D7%A8)" שקיימברידג' אנליטיקה עשתה שימוש עסקי במידע אישי מ[פייסבוק](https://he.wikipedia.org/wiki/%D7%A4%D7%99%D7%99%D7%A1%D7%91%D7%95%D7%A7), שמלכתחילה נאסף על ידי חוקר חיצוני למטרות אקדמיות. בתגובה, אסרה פייסבוק על קיימברידג' אנליטיקה לפרסם בפייסבוק.

האירועים הנ"ל גרמו לפייסבוק לבצע שינוי משמעותי באבטחה. החל מאפריל 2018 לא ניתן לתת הרשאות publish\_actions לאפליקציות – הרשאה שהיה בה צורך להעלאת תמונה לקיר של המשתמש הראשי – כלומר הבית בעזרת API שנקרא FB-RECOG.

נזכיר שמטרות הפרוייקט הוגדרו במרץ 2018.

הניסיון הבא היה להעלות את התמונות לעמוד (page) של פייסבוק, במקום לקיר של משתמש, אך גם פתרון זה נפל! הפעם משום שפייסבוק לא מאפשרים בעזרת graph api לתייג תמונות בעמוד, גם הפעם לצרכי אבטחה ופרטיות.

הפתרון שלבסוף פתר את האתגר בוצע בעזרת Amazon Web Services, ומערכת הRekognition שלה, נפרט עליו עוד בהמשך.

* אתגר נוסף שעמד לפנינו היה שליפת תמונה מתוך streaming של מצלמה והעברתה לשרת לצורך זיהוי. פתרון שלבסוף לא נלקח משום שלא היה יעיל: לעשות THREAD בצד שרת של המצלמה, ופעם בחצי שנייה לעצור אותו, לעשות שליפת תמונה ואז להמשיך את הSTREAMING. הפתרון כמובן לא טוב, אין סיבה לעצור את הזרמת המצלמה בשביל לקחת תמונה, זה עלול לגרום לתקיעות ולפעולות מיותרות של התוכנה.

הפתרון שכן נלקח לבסוף הוא שימוש במצלמה מצד הלקוח, שירות הSTREMING מבוצע ישירות בצד לקוח ע"י JAVASCRIPT, פעם בחצי שנייה הסקריפט ייקח SCREENSHOT מתוך המצלמה, בלי לעצור את הזרמת הנתונים למסך, ויישלח את הלינק הזמני של התמונה לצד שרת. השרת ייגש ללינק הזמני, ישמור את התמונה ולאחר מכן יעבד אותה – בין אם מדובר בזיהוי פנים ובין אם מדובר בזיהוי תנועות.

* אתגר אחר שעמד לפנינו היה זיהוי תנועות ידיים. על מנת לזהות את התנועות השתמשנו באלגוריתם הבא: לקחנו 2 תמונות עוקבות בהפרש של חצי שנייה ועשינו על שתיהן השוואה של תנועה. את הפירוט של האלגוריתם ניתן למצוא בפרק "ארכיטקטורה ומימוש".
* האתגר האחרון שעמד לפנינו היה להריץ את הכל על RASPBERYY PI3. רוב המעברים מחלונות לraspbian עברו בצורה חלקה, אך לא ספריית OPENCV, (עבור זיהוי תנועות ידיים). ספרייה זו איננה ניתנת להורדה למערכת הפעלה זו.

נעשה שימוש במדריך הבא לקומפילציה של הספרייה על הראספברי פאי עצמו:

<https://www.pyimagesearch.com/2016/04/18/install-guide-raspberry-pi-3-raspbian-jessie-opencv-3/>

1. SDKs & APIs

# facebook-jssdk

רץ בצד לקוח ומתממשק עם javascript.

SDK שאנחנו מקבלים מפייסבוק, אחראי על יצירת חלון עם עמוד פייסבוק (או פוסט פייסבוק) באפליקציה שלנו, אשר מכיל את התוכן המבוקש, במקרה זה – אחד העמודים האהובים על המשתמש הנוכחי. נשים לב שהעמוד או הפוסט חייב להיות PUBLIC, מכיוון שאנחנו ניגשים אליו דרך המשתמש הראשי, שייתכן ולא עשה לו לייק.

# AWS - Rekognition

ענן של אמאזון, בתוכו יש API של שירות Rekognition. השירות הינו מאוד חזק ומספק מגוון רחב של אפשרויות זיהוי פנים וניתוח תמונה. השימוש שלנו בAPI בוצע כך:

* בשימוש הראשון של משתמש, עליו להזין את השם משתמש הייחודי שלו בפייסבוק (לא השם כפי שמופיע בעמוד, אלא username), וללחוץ על כפתור "משתמש חדש".
* השם נשלח ביחד עם תמונת משתמש לAPI, ושם נשמר וקטור מידע על תווי הפנים של האדם.
* כעת לAPI יהיה אוסף וקטורים של תווי פנים של משתמשים רבים, וניתן בעזרת שליחת תמונה לAPI, לבקש זיהוי של האדם הנוכחי מול המצלמה.
* אם נמצאה התאמה מדויקת של מעל 90% בין האדם מול המצלמה ואדם באוסף, אפשר להציג את תוכן העמודים האהובים עליו בפייסבוק.

# BEAUTIFULSOUP

ספרייה בפייתון אשר באה לידי שימוש בצד שרת. הספרייה אחראית על PARSING של עמוד HTML אותו היא מקבלת כקלט (URL). במקרה שלנו, אנחנו מזינים לאובייקט של BEAUTIFULSOUP כתובת של עמוד הלייקים של המשתמש הנוכחי, האובייקט ניגש לעמוד, ומפרסר אותו. באחריותנו לקחת את הקובץ XML שיצר הפרסר, ולמצוא בתוכו את האובייקט (CLASS) אשר מכיל בתוכו לינק לעמוד אהוב.

נמצא את כל העמודים האהובים של המשתמש, ונחזיר אותם כמילון של שם + לינק. לאחר מכן ניקח את הלינקים לעמודים האהובים, וגם אותם נשלח לאובייקט נוסף של BEAUTIFULLSOUP על מנת למשוך מהם לינקים לפוסטים בעמודים הנ"ל, ואת הפוסטים נוכל להציג באפליקציה.

הערה חשובה: התוכן שמוחזר מדף HTML מוחזר בתוך הערה, טרם פרסור הHTML, עלינו להוריד את סימני ההערה, על מנת שהBEAUTIFULSOUP יוכל למצוא את תגיות XML הדרושות.

# FLASK

API בפייתון שאחראי על הרצת אתר במתודולוגיה הסטנדרטית של שרת-לקוח. במקרה שלנו השרת הינו מחשב וירטואלי שאנו מפעילים טרם הרצת האפליקציה.

הAPI הוא זה שמאפשר לנו ליצור תקשורת בין הלקוח והשרת, ע"י פנייה של הלקוח ללינקים מוגדרים מראש. לדוגמה פנייה של הלקוח לכתובת: /my\_link"" תגרום לקריאה הבאה של הפונקצייה בצד שרת:

@app.route(**"/my\_link"**, methods=[**'GET'**,**'POST'**])  
**def** getHandGesture():

…

ניתן להעביר מידע מלקוח לשרת בעזרת שליחת מילון data, במבנה ג'ייסוני ביחד עם הבקשה. השרת יקרא את המידע בעזרת גישה לארגומנטים, לדוגמה:

path = request.args.get(**'image\_src'**)

התקשורת בין הצדדים היא בעזרת סטנדרט JSON (ראה נספח). מקור לFLASK:

http://flask.pocoo.org/docs/1.0/

# OPENCV4 + NUMPY

ספריות של PYTHON אשר אחראיות על עיבוד תמונה, בעזרת ספריות אלו אנו מפענחים את תנועות הידיים של המשתמש. לאלגוריתם של הפענוח, ראה פרק "ארכיטקטורה ומימוש".

1. מבט על

האפליקציה עובדת בצורת תכן סטנדרטית של צד לקוח וצד שרת:

Liked pages and posts retrieving

Photo Recognition Flow

Get Post of Current Link

Non-stop **streaming**  
from camera to client-side

Server Side:  
When request with photo URL arrives from client:   
every 10th photo start photo recognition flow with AWS and FB, else compare with previous photo to recognize gesture. Return updates in posts and user to client with JSON.  
If newUser request arrives, send request to AWS for collection addition.

Client Side:  
Each interval of half second, take a camera snapshot, and send link to server.  
Update post window if necessary by response.  
If "new user" button was pressed, send newUser request with name and photo.

Facebook



AWS Rekognition service

Or add photo to collection in case of new user.

Photo Recognition Flow: Server 🡪 AWS 🡪 Server 🡪 FB

1. ארכיטקטורה ומימוש

צד לקוח:

כפי שנאמר תוכנית הMAIN שלנו היינה לולאה אינסופית, אשר נקראת כל חצי שנייה, היא נראית כך:

**var *interval*** = 500; *// miliseconds***var *first\_cycles\_run\_counter*** = 0;  
  
**function** *run\_with\_timer*()  
{  
 *get\_screenshot*();  
 **if** (***first\_cycles\_run\_counter*** > 5) {  
 *get\_next\_post*();  
 } **else** {  
 ***first\_cycles\_run\_counter***++; *// async - takes time for first picture to be uploaded* }  
 *setTimeout*(*run\_with\_timer*, ***interval***);  
}  
  
$(***document***).**ready**(*run\_with\_timer*());

לאחר האתחול הראשוני, אשר לוקח כמה מחזורי קריאה מכיוון שהוא אסינכרוני, הפונקצייה נכנסת למעגל של לקיחת תמונה ושליחת URL שלה לשרת.

עבור מימוש הבקשה לFLASK, נשתמש במבנה הבא עם JSON QUERY:

**function** *get\_next\_post*() {  
 $.getJSON(  
 {  
 **url**: **"/interpretPhoto"**,  
 **data**: {**'image\_src'**: ***document***.getElementById(**'screenshot-img'**).**src**},  
 success: **function** (result) {  
 **if** (result.**status** != **''**) {  
 $(**".fb-post"**).attr(**'data-href'**, result.next\_url);  
  
 **if** (result.**status** == **"new\_post"**) {  
 ***document***.getElementById(**"gesture\_span"**).**textContent** = result.gesture;  
 }  
 **else if** (result.**status** == **"new\_page"**) {  
 ***document***.getElementById(**"page\_name\_span"**).**textContent** = result.page\_name;  
 ***document***.getElementById(**"gesture\_span"**).**textContent** = result.gesture;  
 }  
 **else if** (result.**status** == **"new\_person"**) {  
 ***document***.getElementById(**"gesture\_span"**).**textContent** = **'-'**;  
 ***document***.getElementById(**"person\_name\_span"**).**textContent** = result.person\_name;  
 ***document***.getElementById(**"page\_name\_span"**).**textContent** = result.page\_name;  
 }  
 **else** {  
 ***window***.alert(**'Unrecognized returned status: '** + result.**status**);  
 ***document***.getElementById(**"gesture\_span"**).**textContent** = **'-'**;  
 ***document***.getElementById(**"person\_name\_span"**).**textContent** = **'-'**;  
 ***document***.getElementById(**"page\_name\_span"**).**textContent** = **'-'**;  
 }  
 FB.XFBML.parse();  
 }  
 **else** {  
 ***document***.getElementById(**"gesture\_span"**).**textContent** = **'-'**;  
 }  
  
 }  
 });  
}

ניתן לראות שהפונקצייה בעצם שולחת בקשה לשרת, של "מה להציג הלאה?". הפונקצייה מעבירה בשדה הDATA לינק לתמונה האחרונה שצולמה, אותה השרת יוכל לשמור, וכאשר השרת מגיב הוא צריך להעביר סטטוס.

אם הסטטוס ריק, אזי כלום לא משתנה, ורק צריך לאפס את שדה תנועה מזוהה.

במקרים אחרים, השרת יחזיר את אחת מהתשובות:

* משתמש חדש
* עמוד חדש
* פוסט חדש

בכל אחד מהמקרים הנ"ל, השרת יעביר לינק לפוסט הבא שיש להציג למשתמש, בעזרת facebook plugin.

במידה ונלחץ כפתור "new user", אז צד הלקוח ייקרא לינק נמתאים בצד שרת על מנת להוסיף אותו לרשימת המשתמשים, וגם מיד להציג את העמודים האהובים עליו:

function new\_user() {

$.getJSON(

{

url: "/newUser",

data: {'image\_src': document.getElementById('screenshot-img').src,

'user\_name': document.getElementById('new\_user\_input').value},

success: function (result) {

if (result.status != '') {

$(".fb-post").attr('data-href', result.next\_url);

if (result.status == "new\_person") {

document.getElementById("gesture\_span").textContent = '-';

document.getElementById("person\_name\_span").textContent = result.person\_name;

document.getElementById("page\_name\_span").textContent = result.page\_name;

}

else {

window.alert('Unrecognized returned status: ' + result.status);

document.getElementById("gesture\_span").textContent = '-';

document.getElementById("person\_name\_span").textContent = '-';

document.getElementById("page\_name\_span").textContent = '-';

}

FB.XFBML.parse();

}

else {

document.getElementById("gesture\_span").textContent = '-';

}

}

});

}

פונקציית הזרמת תמונה ללא הפסקה (STREAMING):

***navigator***.**mediaDevices**.getUserMedia(***constraints***).then(*handleSuccess*).catch(*handleError*);  
**function** *handleSuccess*(stream) {  
 ***video***.**srcObject** = stream;  
}  
**function** *handleError*(error) {  
 ***console***.error(**'Error: '**, error);  
}

פונקציית לקיחת SNAPSHOT רגעי מהמצלמה:

**function** *get\_screenshot*()  
{  
 ***canvas***.**width** = ***video***.**videoWidth**;  
 ***canvas***.**height** = ***video***.**videoHeight**;  
 ***canvas***.getContext(**'2d'**).drawImage(***video***, 0, 0);  
 ***img***.**src** = ***canvas***.toDataURL(**'image/png'**);  
}

צד שרת:

צד שרת נכתב בפייתון, בגלל נוחות התממשקות עם ספרייה קיימת FBRECOG אשר כתובה בפייתון (תזכורת – זוהי ספרייה לזיהוי פנים בתמונה ע"י פייסבוק), וכמו כן שימוש בFLASK, שזהו הAPI לתקשורת עם צד לקוח.

הערה חשובה: כפי שהוסבר בפרק "אתגרים בפיתוח" לבסוף פתרון FB-RECOG לא נלקח משום שפייסבוק חסמו אפשרות להעלאת תמונה, אך עבור פרוייקט זה הדבר לא היווה בעיה.

שירות הRekognition של AWS גם הוא עובד עם פייתון, וההתמשקות איתו הייתה מהירה וקלה.

השרת מכיל פונקציית תגובה לבקשה של צד לקוח לזיהוי תמונה:

@app.route(**"/interpretPhoto"**, methods=[**'GET'**])  
**def** interpret\_photo():

…

ונשים לב שניתן להעביר מידע מהלקוח לשרת בצורה הבאה:

path = request.args.get(**'image\_src'**) *# image link passed from javascript*

זוני בעצם הפונקציה הראשית של צד שרת. פונקציה זו אחראית על השוואת תמונות עוקבות לצורך זיהוי תמונות ידיים, ושליחת בקשה לפייסוק לזיהוי פנים בעזרת פונקציית עזר: \_get\_person\_name, אם עומד מול המצלמה משתמש חדש, היא תפנה לאלגוריתם זיהוי עמודים מועדפים (יוסברבהמשך) .

הפונקציה תחזיר לצד לקוח עדכון עם הפוסט הבא להצגה + מידע על המשתמש והעמוד.

פונקציות עזר:

**def** \_get\_person\_name()

**def** \_interpret\_hand\_gesture(prev\_photo\_name, next\_photo\_name)

**def** \_create\_liked\_posts\_list(index\_to\_create\_for)

**def** \_create\_liked\_pages\_list()

**def** \_get\_hand\_gesture(img\_name\_first, img\_name\_second)

**def** \_compare\_images\_for\_gesture(first\_avg, second\_avg)

**def** \_login(session, email, password)

וכמובן פונקציית MAIN שמריצה את שרת הFLASK הוירטואלי:

**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 app.run(debug=**True**)

זיהוי בן אדם:

על מנת לזהות בן אדם, עלינו קודם כל ליצור עבורו וקטורי של תווי פנים בשירות Rekognition. הדבר מתבצע כאשר נקראת הפונקצייה הבאה מצד לקוח:

@app.route(**"/newUser"**, methods=[**'GET'**])  
**def** new\_user():  
 **global** liked\_page\_index  
 **global** liked\_post\_index  
 **global** curr\_friend\_name  
  
 path = request.args.get(**'image\_src'**) *# image link passed from javascript* username = request.args.get(**'user\_name'**) *# new user name from javascript* res\_dict = {**'status'**: **''**}  
 urllib.request.urlretrieve(path, **"photo\_for\_new\_user.jpg"**)  
  
 client = boto3.client(**'rekognition'**)  
 colId = **"SmartSocNet"** *#c = client.create\_collection(CollectionId=colId)* **with** open(**"photo\_for\_new\_user.jpg"**, **"rb"**) **as** imgf:  
 img = imgf.read()  
 indr = client.index\_faces(CollectionId=colId, Image={**'Bytes'**: img}, ExternalImageId=username, MaxFaces=1, )  
  
 curr\_friend\_name = username  
 liked\_page\_index = 0  
 \_create\_liked\_pages\_list()  
 next\_page\_name = pages\_to\_show[liked\_page\_index][**'name'**]  
 liked\_post\_index = 0  
 next\_post = pages\_to\_show[liked\_page\_index][**'posts'**][liked\_post\_index]  
 res\_dict[**'status'**] = **'new\_person'** res\_dict[**'person\_name'**] = curr\_friend\_name  
 res\_dict[**'page\_name'**] = next\_page\_name  
 res\_dict[**'next\_url'**] = next\_post  
  
 **return** jsonify(res\_dict)

נזכור שהפונקציה נקראת בלחיצה על כפתור "new user" , וצד הלקוח מעביר לשרת שם שהוזן ותמונה עדכנית. פונקצייה זו תכניס את האדם לאוסף: colId = **"SmartSocNet"**

ע"י הפקודה: indr = client.index\_faces(CollectionId=colId, Image={**'Bytes'**: img}

כעת נוכל להשתמש בפונקציית זיהוי משתמש הבאה, כדי לזהות את האנשים הקיימים במערכת:

**def** \_get\_person\_name():  
 **global** curr\_friend\_name  
 **global** colId  
friend\_name = curr\_friend\_name  
 **try**:  
client = boto3.client(**'rekognition'**)  
 **with** open(**"photo\_for\_recognition.jpg"**, **"rb"**) **as** imgf:  
 img = imgf.read()  
 inds = client.search\_faces\_by\_image(CollectionId=colId, Image={**'Bytes'**: img}, MaxFaces=1)  
  
 **if** (len(inds) > 0) **and** (inds[**'FaceMatches'**][0][**'Similarity'**] > 90):  
 friend\_name = inds[**'FaceMatches'**][0][**'Face'**][**'ExternalImageId'**]  
  
 **except** Exception **as** err:  
 print(err)  
 print(**'Error occurred, please check the token, and verify you are connected.'**)  
  
 **return** friend\_name

הזיהוי מתבצע עם הפקודה:

inds = client.search\_faces\_by\_image(CollectionId=colId, Image={**'Bytes'**: img}, MaxFaces=1)

ומתקבל בחזרה מערך של זיהויים עם הסתברויות זיהוי נכונות.

אנו בוחרים את האדם שיש לו את הסתברות הזיהוי הרבה ביותר, בתנאי שהיא מעל 90%.

זיהוי עמודים מועדפים והעברתם לשרת:

זיהוי עמודים מועדפים של החבר יתבצע בעזרת הפונקצייה: \_create\_liked\_pages\_list(). הפונקצייה מכירה את האדם שכרגע מול האפליקציה (הוא כבר זוהה), ולכן היא יכולה להתחבר כמשתמש הראשי לפייסבוק, ולהגיע דרך URL לעמודים הלייקים של המשתמש הנוכחי (בעזרת \_login):

likes\_url = **'https://www.facebook.com/'** + dotted\_name\_of\_curr\_friend +**'/likes?lst=1198688678%3A843054236%3A1535896155'**response = session.get(likes\_url, cookies=cookies, allow\_redirects=**False**)

מכאן נשתמש בספרייה של BEAUTIFUSOUP שעליה הסברנו קודם, על מנת להוציא מעמוד הלייקים את השמות והלינקים של העמודים והפוסטים האהובים, ולבסוף לשלוח לצד לקוח את הפוסט הבא להצגה.

זיהוי תנועות ידיים:

עלינו לזהות 4 תנועות ידיים: למעלה, למטה, משיכה לימין ומשיכה לשמאל. הדבר מתבצע בעזרת הפונקצייה: \_get\_hand\_gesture, אשר מקבלת 2 תמונות, מפעילה על כל אחת מהן את האלגוריתם הבא, ולאחר מכן שולחת את התוצאות ל: \_compare\_images\_for\_gesture.

האלגוריתם:

* תמונה מקורית:



* הפוך את התמונה לגווני שחור – לבן:



* טשטש את התמונה מעט על מנת להמנע מרעשים:



* קיטוב צבעי התמונה לשחור אם הצבע בין 0 ל100, וללבן אם הצבע בין 101-255:



* וכעת ניתן לחשב את ממוצע הצבע הלבן בתמונה, למשל: [285, 261].
* \_compare\_images\_for\_gesture מקבלת 2 ממוצעים של צבעים לבנים ב2 תמונות. נזכור שהתמונות הן במרווח זמן של חצי שנייה. נוכל לראות לאיזה כיוון זז ממוצע הצבע הלבן בתמונה השנייה. לפי כיוון התזוזה המקסימלית נדע אם תנועת היד הייתה בכיוון מעלה \ מטה \ ימינה \ שמאלה.

1. סיכום

ניתן לראות שהתממשקות עם פייסבוק היא אפשרית, עם עמידה באתגרי אבטחה. זיהוי פנים ותנועות ידיים מבוצע בצורה טובה אפילו עם מצלמה פשוטה יחסית.

המסקנה היא שניתן ליצור בית חכם ונוח לשימוש כפי שהוגדר ביעדים.

1. ביבליאוגרפיה
2. נספחים

JSON:

מתוך ויקיפדיה: <https://he.wikipedia.org/wiki/JSON>

**JSON** ראשי תיבות של **J**ava**S**cript **O**bject **N**otation) ) הוא [פורמט](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A8%D7%9E%D7%98_%D7%A7%D7%95%D7%91%D7%A5) טקסטואלי, הקריא לאדם, המיועד להעברת מבני מידע המורכבים מזוגות של מפתח-ערך. השימוש העיקרי של הפורמט הוא להעברת מידע בין שרת לצרכן כתחליף לפורמט XML.