

בית ספר: תיכון חדרה

שם הפרויקט: SongFeeler

שם התלמיד : גל פחימה

ת.ז. : 323973446

מנחה: אלון פלומן

תאריך הגשה: 26.05.2022

תוכן

[מבוא 6](#_Toc105881677)

[תיאור כללי 6](#_Toc105881678)

[תהליך המחקר 6](#_Toc105881679)

[הסיבה לבחירת הפרויקט 6](#_Toc105881680)

[הבעיה עימה התמודדתי 6](#_Toc105881681)

[סוגי משתמשים / קהל היעד 7](#_Toc105881682)

[יעדים 7](#_Toc105881683)

[צד שרת המספק 7](#_Toc105881684)

[צד לקוח המספק 7](#_Toc105881685)

[המוטיבציה לעבודה 7](#_Toc105881686)

[הצורך שעליו עונה הפרויקט והפתרון שהפרויקט בא לתת 7](#_Toc105881687)

[אילוצים ודרישות 8](#_Toc105881688)

[אילוצי זמן 8](#_Toc105881689)

[אילוצי תוכנה וידע 8](#_Toc105881690)

[אילוצי למידה 8](#_Toc105881691)

[פתרונות קיימים 8](#_Toc105881692)

[המלצת שירים לפי רגשות או טעם 8](#_Toc105881693)

[זיהוי רגש בפנים 8](#_Toc105881694)

[דרישות מערכת 8](#_Toc105881695)

[שרת 8](#_Toc105881696)

[אפליקציה 8](#_Toc105881697)

[פיצ'רים ותהליכים עיקריים 9](#_Toc105881698)

[ציר טכנולוגי 9](#_Toc105881699)

[מדריך התקנת סרבר 10](#_Toc105881700)

[שלב 1: 10](#_Toc105881701)

[שלב 2: 10](#_Toc105881702)

[שלב 3: 10](#_Toc105881703)

[שלב 4: 10](#_Toc105881704)

[שלב 5: 11](#_Toc105881705)

[שלב 6: 11](#_Toc105881706)

[מדריך התקנת אפליקציה 12](#_Toc105881707)

[שלב 1: 12](#_Toc105881708)

[נעביר את קובץ ה APK דרך האפליקציה WhatsApp 12](#_Toc105881709)

[שלב 2: 12](#_Toc105881710)

[נלחץ על כפתור ההורדה ומשם נעקוב אחרי ההדרכה של הטלפון 12](#_Toc105881711)

[שלב 3: 12](#_Toc105881712)

[ארכיטקטורת הפרויקט 13](#_Toc105881713)

[הצגת הפתרון המוצע 13](#_Toc105881714)

[רכיבים 14](#_Toc105881715)

[Face Recognition 14](#_Toc105881716)

[Server 14](#_Toc105881717)

[DB (מסד נתונים) 14](#_Toc105881718)

[App 14](#_Toc105881719)

[תיאור זרימת המידע במערכת 15](#_Toc105881720)

[מבני הנתונים בפרויקט 15](#_Toc105881721)

[רשימה - List 15](#_Toc105881722)

[מערך בתים – bytearray 15](#_Toc105881723)

[מילון - Dictionary 15](#_Toc105881724)

[טאפל - Tuple 15](#_Toc105881725)

[ארכיטקטורת רשת 16](#_Toc105881726)

[תיאור פרוטוקולי תקשורת 16](#_Toc105881727)

[פירוט על הטכנולוגיות 17](#_Toc105881728)

[Python 17](#_Toc105881729)

[NumPy/Keras 17](#_Toc105881730)

[Flutter 17](#_Toc105881731)

[SQLITE 17](#_Toc105881732)

[Neural network 17](#_Toc105881733)

[Socket 17](#_Toc105881734)

[Google Colab 17](#_Toc105881735)

[OpenCV 17](#_Toc105881736)

[Auto-py-to-exe 17](#_Toc105881737)

[TensorFlow 17](#_Toc105881738)

[טיפול בחריגות 18](#_Toc105881739)

[טיפול בחריגות בצד השרת: 18](#_Toc105881740)

[טיפול בחריגות בצד הלקוח: 18](#_Toc105881741)

[Open sources 18](#_Toc105881742)

[Use Case 19](file:///C:\Users\Gal%20Pahima\Desktop\SongFeeler\תיק%20פרויקט.docx#_Toc105881743)

[מדריך משתמש 20](#_Toc105881744)

[Screen Flow Diagram 20](file:///C:\Users\Gal%20Pahima\Desktop\SongFeeler\תיק%20פרויקט.docx#_Toc105881745)

[Server UML 21](#_Toc105881746)

[Sequence 22](#_Toc105881747)

[בסיס הנתונים 24](#_Toc105881748)

[חולשות מטופלות 24](#_Toc105881749)

[מסכים 25](#_Toc105881750)

[מסך ספלאש 25](#_Toc105881751)

[מסך שגיאת התחברות 25](file:///C:\Users\Gal%20Pahima\Desktop\SongFeeler\תיק%20פרויקט.docx#_Toc105881752)

[מסך התחברות 26](#_Toc105881753)

[מסך הרשמה 27](#_Toc105881754)

[מסך בית 28](#_Toc105881755)

[מסך טעינה 29](#_Toc105881756)

[מסך שגיאת שרת 29](#_Toc105881757)

[מסך תוצאה 30](#_Toc105881758)

[מדריך למפתח 31](#_Toc105881759)

[main.py 31](#_Toc105881760)

[server.py 31](#_Toc105881761)

[db\_manger.py 33](#_Toc105881762)

[Client\_handler.py 33](#_Toc105881763)

[מסך התחברות והירשמות 37](#_Toc105881764)

[Helper.dart 38](#_Toc105881765)

[login\_signup.dart 38](#_Toc105881766)

[menu.dart 38](#_Toc105881767)

[socket.dart 39](#_Toc105881768)

[קוד המכונה 41](#_Toc105881769)

[רפלקציה 43](#_Toc105881770)

[חוויה כללית מתהליך העבודה 43](#_Toc105881771)

[מה קיבלתי מהפרויקט 43](#_Toc105881772)

[כלים שאקח להמשך הדרך 43](#_Toc105881773)

[אתגרים 43](#_Toc105881774)

[מסקנות 43](#_Toc105881775)

[מה הייתי עושה אחרת לו הייתי מתחיל היום 43](#_Toc105881776)

[מה היה קורה אחרת אם העבודה הייתה יותר יעילה עבורי 43](#_Toc105881777)

[רעיון לפיתוח עתידי 43](#_Toc105881778)

[ביבליוגרפיה 44](#_Toc105881779)

# מבוא

## תיאור כללי

פרויקט הגמר שביצעתי נקרא SongFeeler. הפרויקט משתמש בלמידת על מנת לזהות רגש בפנים של המשתמש. לאחר זיהוי הרגש יש התאמה בין הרגש של המשתמש לשיר מסוים לפי אותו רגש. לכן, ניתן לומר שהתוכנה מרגישה את השיר ומכאן שמה.

## תהליך המחקר

הפרויקט דרש ממני להרחיב את הידע שלי בנושאים שלא התעסקתי בהם קודם:

* למידת מכונה – למדתי לעומק איך לפתח מכונה לומדת בשפת התכנות Python באמצעות הספרייה TensorFlow, כדי ללמוד את הנושא לעומק עברתי קורס באתר Coursera, הקורס נקרא Machine Learning by Stanford.
* GUI – מעולם לא התעסקתי עם ממשק משתמש לפני כן, לכן שהחלטתי ללמוד איך לפתח אפליקציה בחרתי בשפה התכנות Flutter, שפה חדשה יחסית אך עונה על הצרכים, על מנת ללמוד את השפה צפיתי בסרטוני YouTube ונעזרתי באתר האינטרנט Stack Overflow.

## הסיבה לבחירת הפרויקט

בחרתי בתחום הפרויקט מכיוון שתחום המכונה הלומדת, הוא תחום שעניין אותי כבר זמן רב, ראיתי בפרויקט הזדמנות ללמידת הנושא והזדמנות להרחבת הידע שלי. לכן, הלכתי לכיוון הלומדה. נושא הפרויקט הגיע כאשר שמעתי שירים באפליקציה YouTube Music, אחד הפיצ'רים באפליקציה מאפשר לבחור שירים לפי מצב רוח, אך, לא מאפשר זיהוי רגש בפנים. לכן עלה בראשי רעיון לשלב את המכונה הלומדת על מנת לזהות רגשות ולהתאים לפיהם שירים כמו ב YouTube Music.

## הבעיה עימה התמודדתי

כאשר התחלתי את הפרויקט נתקלתי במספר בעיות עיקריות:

* מציאת Dataset מתאים – על מנת לאמן את המכונה צריך Dataset מתאים המכיל מספיק תמונות על מנת לאמן את המכונה, סוג המודל שבחרתי מצריך Dataset גדול על מנת להגיע לדיוק מספק.
* קבלת הודעות מצד המשתמש – Flutter היא שפה חדשה יחסית ולכן אין עליה הרבה פרטים בנושא ה Sockets. אז היה צריך פתרון יצירתי ל

## סוגי משתמשים / קהל היעד

אנחנו מקווים לייעד את התוכנה לכלל השוק, מכיוון שמוזיקה לא תלויה בגיל, ילדים, מבוגרים וצעירים כאחד שומעים מוזיקה, לכן הפרויקט שלנו מיועד לכל מי שרוצה לשמוע מוזיקה שמתאימה למצב רוחו.

## יעדים

### צד שרת המספק

* הרשמה והוספה של משתמשים למסד נתונים.
* זיהוי רגש בפנים והתאמתו לשיר.

### צד לקוח המספק

* מסכי טעינה.
* מסכי שגיאה.
* מסך בית.
* מסכי הרשמה והתחברות.
* בדיקת נתוני סיסמא ואימייל לפני שליחתם.

## המוטיבציה לעבודה

הדבר העיקרי אשר הניע אותי לעשות את העבודה, הוא הרצון ללמוד דברים חדשים במהלך עשיית הפרויקט וללמוד על כמה שיותר תחומים. בנוסף לכך, המוטיבציה באה גם מהרצון לבצע הפרויקט על הצד הטוב ביותר כדי לצבור ניסיון תכנותי וידע חדש שיעזור לי בהמשך הדרך.

## הצורך שעליו עונה הפרויקט והפתרון שהפרויקט בא לתת

מוזיקה היא חלק בלתי נפרד מהחיים של רובנו, אנחנו שומעים אותה במקלחת, במסיבות, במכונית, במסעדות, ואני יכול להמשיך לתת דוגמאות עד מחר. המוזיקה שאנחנו שומעים מותאמת למצב הרוח שלנו, אם אנחנו שמחים נשמע שירים קצביים, אם אנחנו עצובים נשמע שירים יותר עצובים וכו'...

יש פתרונות קיימים של YouTube ושל Spotify שמציעים לך שירים לפי המצב רוח, אך לא מאפשרים זיהוי של מצב הרוח. אז רציתי לתת אפשרות לכל אלה שרוצים לשמוע מוזיקה לפי הרגש שלהם או לכאלה שסתם רוצים להשתעשע עם אפליקציה נחמדה.

## אילוצים ודרישות

### אילוצי זמן

* שבעה חודשים לפיתוח הפרויקט.
* תלמיד בתוכנית מגשימים.

### אילוצי תוכנה וידע

* ידע מעמיק בשפת Python על מנת לפתח שרת Multi-Threaded.
* ידע מעמיק בתחום המכונה הלומדת.
* ידע מעמיק בשפת Flutter על מנת לפתח את האפליקציה.

### אילוצי למידה

* למידה עצמית – למידה מסרטונים, פורומים, קורסים למיניהם וכו'...

## פתרונות קיימים

### המלצת שירים לפי רגשות או טעם

* Spotify.
* YouTube Music.

### זיהוי רגש בפנים

* [Measure your emotions - FaceReader | Noldus](https://www.noldus.com/facereader/measure-your-emotions)
* [Use Machine Learning Technology | Safe Software](https://www.safe.com/demos/emotion-detector/)

## דרישות מערכת

### שרת

* השרת רץ בסביבת Windows.
* עליו להיות מחובר לאותה רשת פנימית עם האפליקציה.
* שטח אחסון בגודל 2 GB לאחסון השרת ומסד הנתונים שייווצר.

### אפליקציה

* האפליקציה רצה בסביבת Android.
* עליה להיות מחוברת לאותה רשת פנימית עם השרת.
* שטח אחסון בגודל 50 MB לאחסון האפליקציה.

## פיצ'רים ותהליכים עיקריים

1. זיהוי רגש בפנים של בן אדם - מודל זה יהיה מסוג רשת נוירונים ( מודל CNN ) ומטרתו תהיה לזהות רגש בפניו של האדם. ה Input שלו יהיה תמונה של פנים וה Output תהיה רגש מסוים. מודל זה ישמש כ - Input למודל השלישי מה שיאפשר לנו להציע את השיר בהתבסס על הרגש של הבן אדם.
2. ממשק גרפי למשתמש - GUI מתאים אפליקציה לנוחות המשתמש בו יוכל להירשם ויוכל להעלות את התמונה ולקבל תשובה בצורה נוחה.
3. הגנה על הסרבר - יכולת לדעת להתמודד עם מתקפות פשוטות כגון: SQL injection, DDOS, מכיוון שאנחנו שומרים בסרבר שלנו מידע על אנשים. כלומר הגנה על הסרבר הכרחית.
4. הרשמה והתחברות – על מנת לממש מודל שרת-לקוח, פיצ'ר זה הכרחי. לכן כל לקוח יכול להירשם ולהתחבר לאפליקציה. פרטי ההרשמה או פרטי ההתחברות ירשמו בתוך מסד הנתונים. כמובן שבוצעו בדיקות על המידע הנכנס מול מסד הנתונים.
5. הכוונה לאתרים לשמוע מהם שירים – לאחר שהשרת ישלח למשתמש את השיר שהוא מצא מתאים, תהיה אפשרות לנגן את השיר מ YouTube.

## ציר טכנולוגי

* הכנת Dataset או מציאת אחד לאימון המכונה. (יומיים).
* התקנה ספריות נדרשות TensorFlow, NumPy, Keras, cv2 . (שעה).
* ארגון Dataset - הוצאת מידע לא נחוץ, הוספת מידע, שינוי העמודות וכו' (חצי יום).
* קריאת מאמרים בנושא מכונה לומדת. (ארבעה ימים).
* צפייה בקורס על מכונה לומדת על מנת לממש את המודל בצורה הנכונה ביותר. (10 שבועות).
* בניית המודל. (שבוע עד שבועיים).
* עבודה ולמידה של Flutter (שבוע).
* התקנת סביבה עבודה (שלוש שעות).
* ממשק גרפי למשתמש. (שלוש שבועות).
* חיפוש שירים וקישורים. (שעתיים).
* הכנת השרת כולל הכנת Database. (שבוע וחצי).

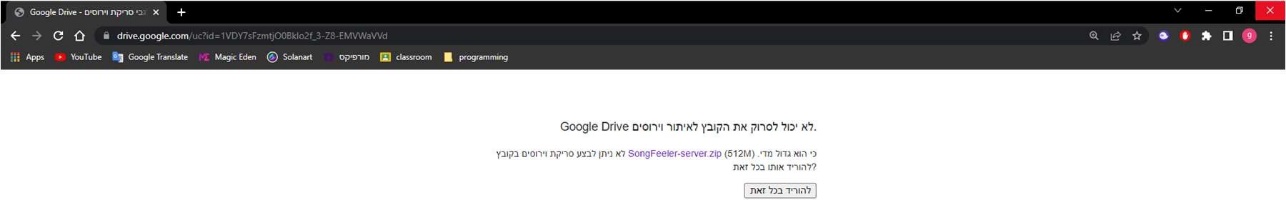
## מדריך התקנת סרבר

### שלב 1:

מורידים את קבצי השרת מהקישור: <https://tinyurl.com/sever-download-sf>.

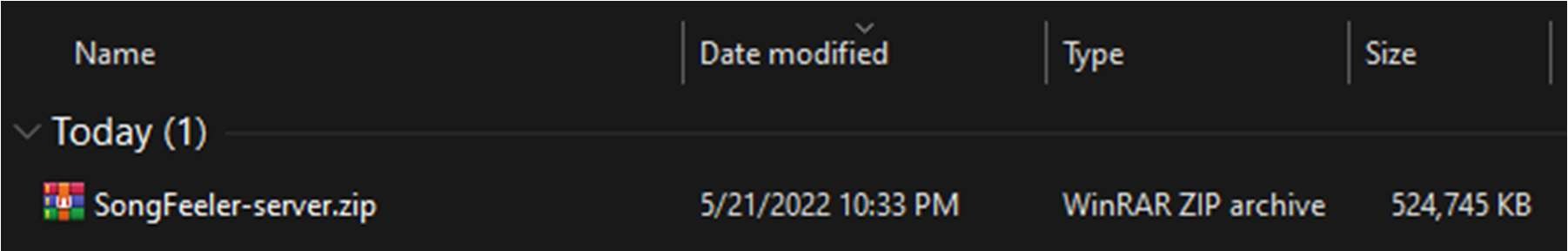
### שלב 2:

ייפתח חלון בגוגל ונלחץ על להוריד בכל זאת



:3 שלב

### שלב 3:

נפתח את תיקיית ההורדות ונעביר את הקובץ למקום נוח כמו מסמכים.

### שלב 4:

תמונה שמכילה טקסט, צג, צילום מסך

התיאור נוצר באופן אוטומטילאחר שהעברנו נלחץ מקש ימיני על הקובץ, אחרי שייפתח החלון נלחץ Extract All…

### שלב 5:

ייפתח החלון בא ונלחץ Extract

תמונה שמכילה טקסט

התיאור נוצר באופן אוטומטי

### שלב 6:

תמונה שמכילה טקסט

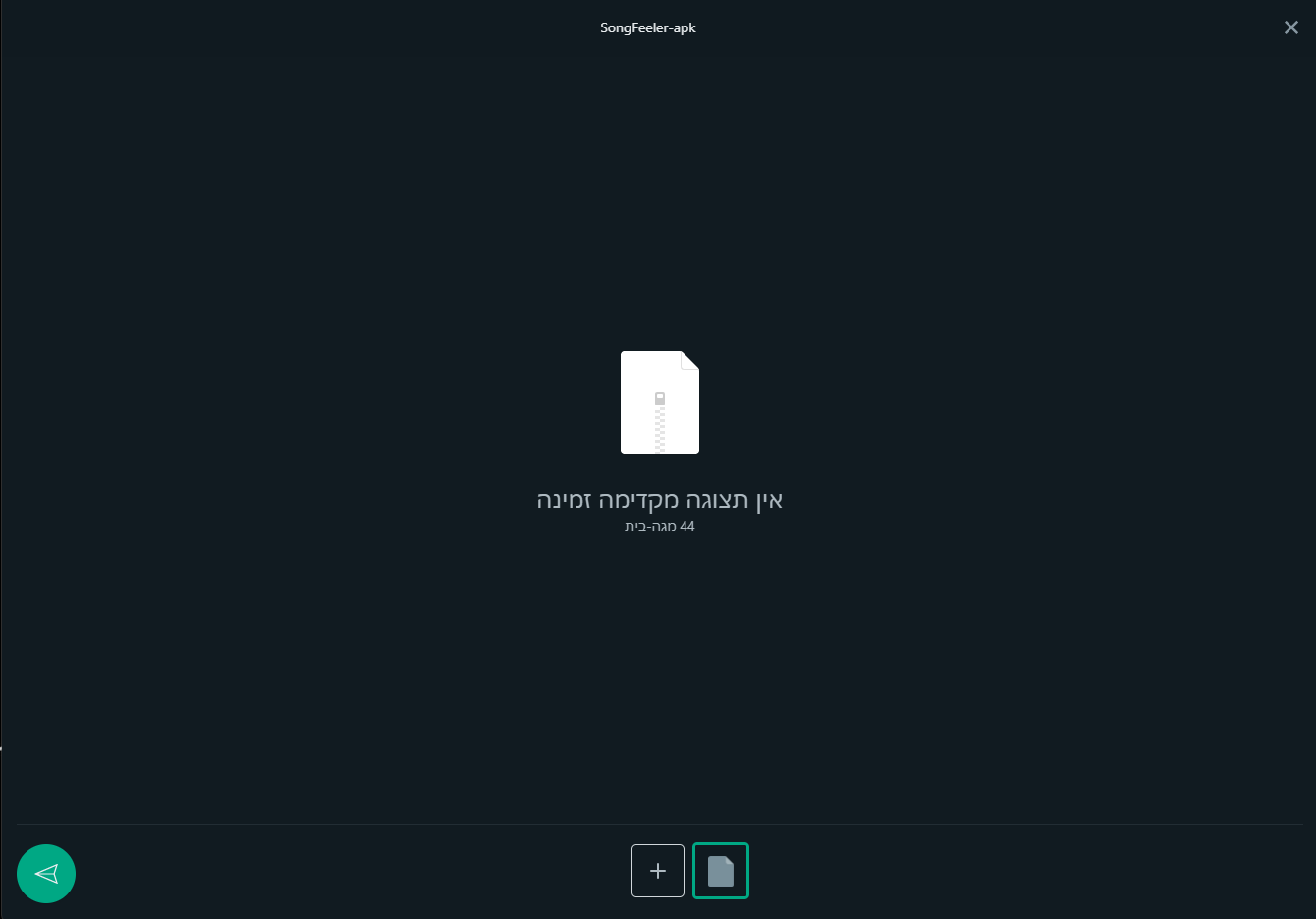
התיאור נוצר באופן אוטומטינחכה לסיום הפעולה ואז נוכל להיכנס לתיקייה לגלול טיפה עד שנראה main.exe נלחץ פעמיים על מנת להריץ וזהו הכל מוכן 😊

## מדריך התקנת אפליקציה

### שלב 1:

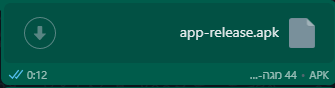
## נעביר את קובץ ה APK דרך האפליקציה WhatsApp

## 



### שלב 2:

## נלחץ על כפתור ההורדה ומשם נעקוב אחרי ההדרכה של הטלפון



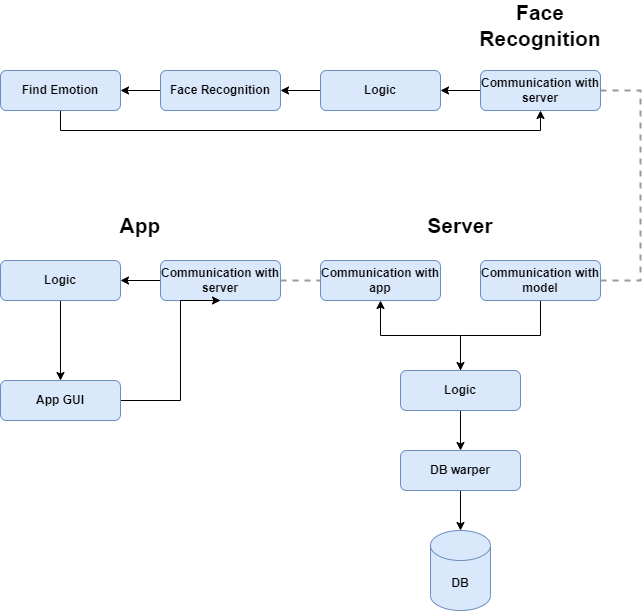
### שלב 3:

ללחוץ על סמל האפליקציה שירדה במסך הבית ולהנות 😊

# ארכיטקטורת הפרויקט

## הצגת הפתרון המוצע

הפתרון המוצע הוא התוכנה שלי SongFeeler. התוכנה כוללת ממשק משתמש משעשע בו המשתמש יכול לגשת בצורה נוחה לשרת, המשתמש יכול להעלות תמונה, התמונה תשלח לשרת, השרת יזהה רגש ויתאים לו שיר, ובסופו של דבר המשתמש יקבל תשובה בממשק משתמש בצורה יפה עם אפשרות לשמוע את השיר ב YouTube. הליבה הטכנולוגית של הפרויקט היא היכולת לזהות רגש בפנים של בן אדם וההתאמה של השיר לפי סגנון המוזיקה שלו והרגש אותו התוכנה זיהתה.



## רכיבים

### Face Recognition

* Communication withserver – רכיב האחראי על התקשורת מול השרת כך שיקבל ממנו את המידע ויעביר אותו ל-Logic.
* Logic רכיב לוגיקה – רכיב האחראי על עיבוד הנתונים המתקבלים בתקשורת, בדיקת הנתונים והעברתם למודל למציאת פנים וזיהוי רגש.
* Face Recognition – רכיב האחראי על זיהוי פנים בתמונה.
* Find Emotion – רכיב האחראי על זיהוי רגש בפנים שזוהו.

### Server

* Communication withapp – רכיב האחראי על התקשורת מול האפליקציה כך שיקבל ממנו את המידע ויעביר אותו ל-Logic.
* Communication withmodel – רכיב האחראי על התקשורת מול המודל כך שיקבל ממנו את המידע ויעביר אותו ל-Logic.
* Logic רכיב לוגיקה – רכיב האחראי על עיבוד הנתונים והבקשות המתקבלים בתקשורת, אימות או עדכון שלהם מול רכיב בסיס הנתונים, והעברת הבקשות למענה המתאים.
* DB Wrapper רכיב בסיס נתונים – רכיב האחראי לשלוף, להוסיף ולעדכן נתונים מול בסיס הנתונים (ניתן להניח שהטבלאות כבר קיימות בו ונוצרו מראש). רכיבים אחרים (כמו לוגיקה) מתקשרים מולו על מנת לעבוד באופן מסודר, בטוח ויעיל מול בסיס הנתונים של התוכנה.

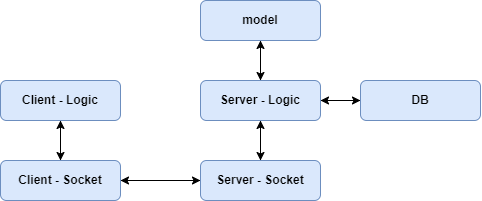
### DB (מסד נתונים)

* הרכיב האחראי על שמירת הנתונים בנוגע לנתונים על המשתמשים.
* רכיב ה-DB יתוכנן בסביבת עבודה DB browser בשפה sqlite3++.

### App

* Communication with Server – רכיב האחראי על התקשורת מול השרת כך שיקבל ממנו את המידע ויעביר אותו ל-Logic.
* Logic רכיב לוגיקה – רכיב האחראי על עיבוד הנתונים והבקשות המתקבלים בתקשורת ועיבודם במידת הצורך כך שיהיה אפשר להציג אותם למשתמש באפליקציה.
* App GUI - אפליקציה דרכה המשתמש יוכל לגשת לתוכנה שלנו ולתקשר עם השרת.

## תיאור זרימת המידע במערכת



## מבני הנתונים בפרויקט

### רשימה - List

רשימה היא טיפוס שיכול להכיל בתוכו טיפוסים שונים כמו מספרים, מחרוזת, תמונה, רשימות ועוד. רשימה ניתנת לשינוי, ולכן השתמשתי במבנה נתונים זה כאשר ידעתי שאצטרך לשמור מספר נתונים ביחד ואצטרך להוסיף או לשנות אותם.

### מערך בתים – bytearray

מערך בתים הוא מערך רגיל לכל דבר, אך מכיל רצף מספרים (0 <= x < 256 ). המערך ניתן לשינוי, ולכן השתמשתי במבנה נתונים זה לקבלת ההודעות. העברתי תמונות ב Socket לכן צריך מבנה נתונים שיכול להכיל את כל הבתים.

### מילון - Dictionary

מילון הוא טיפוס באמצעותו משייכים מפתח לערך, לכן השתמשתי בו כאשר רציתי לשמור כמה ערכים שכל אחד מהם רציתי לתאר, לדוגמה כאשר רציתי לשמור את השירים והרגשות בצורה נוחה לגישה השתמשתי ברגשות כמפתח ובשירים כערך.

### טאפל - Tuple

בפיתון ישנו טיפוס נתונים בסיסי נוסף שנקרא tuple, ומהווה רשימה בלתי משתנה (immutable list). מרגע יצירת רשימה כזו, לא ניתן עוד לשנות את תכנה בשום צורה – אך ניתן לגשת אליו למטרת קריאה.

## ארכיטקטורת רשת

### תיאור פרוטוקולי תקשורת

בצד שרת כל המידע היוצא והנכנס עובר באמצעות ספריית ה Sockets אשר מסופקת ב Python. בצד הלקוח המידע היוצא והנכנס עובר באמצעות ספריית ה Sockets אשר מסופקת ב Flutter.

החיבור בתוכנה הוא חיבור TCP בין השרת והלקוח, השרת שומר על מצב הלקוח מה שהופך אותו לשרת Stateful. כלומר, לאורך כל השיחה בין השרת ללקוח, הם משתמשים באותו Socket להעברת ההודעות. ההודעות נשלחות בפורמט מסוים על מנת להפוך את התקשורת בין השרת ללקוח לקלה.

בתחילת השיחה נפתח קשר ה-TCP ע"י לחיצת יד משולשת ( Three way handshake ) ע"י שליחת 3 הודעות בין הלקוח לשרת. בראשית החיבור הלקוח שולח הודעת SYN לשרת, השרת מחזיר לו הודעת ACK-SYN ,ולסיום הלקוח מחזיר לו ACK. לחיצת היד ממומשת בתוך הפרוטוקול ולכן אין מימוש שלה.

פורמט ההודעות הנשלחות על ידי הלקוח והשרת נראה כך:

|  |  |  |
| --- | --- | --- |
| Message | Message Length | Code |
| מיוצגת על ידי x ביטים | מיוצג על ידי 4 ביטים | מיוצג על ידי ביט |

הקוד הנשלח על ידי הלקוח אינו קבוע, מכיוון שכל קוד מייצג פעולה אחרת.

1 – התחברות

2 – הרשמה

3 – זיהוי רגש והתאמת שיר

4 – התנתקות

הקוד הנשלח על ידי השרת קבוע, הקוד מסמן את תחילת ההודעה של השרת.

## פירוט על הטכנולוגיות

### Python

שפת התכנות Python היא אחת מהשפות הנפוצות ביותר לתכנות מכונה לומדת. לכן אנחנו מתכוונים להשתמש ב Python כדי לתכנת את המכונה.

### NumPy/Keras

טכנולוגיות המציעות מבני נתונים ופעולות לעיבוד מבני נתונים מספריים.

### Flutter

Flutter היא שפה המיועדת לבניית ממשק משתמש. שפת התכנות Flutter מבוססת על שפת התכנות Dart ועושה שימוש בהרבה מהתכונות המתקדמות ביותר של השפה.

### SQLITE

SQLite היא שפת תכנות המיועדת לעבודה מול מסד נתונים, השפה מאפשרת פקודות העוזרות לארגן, להכניס, להוציא ועוד... נתונים ממסד הנתונים.

### Neural network

רשת נוירונים היא סדרה של אלגוריתמים השואפים לזהות מערכות יחסים בסיסיות במערך נתונים באמצעות תהליך המחקה את אופן פעולתו של המוח האנושי. לכן, אנחנו הולכים להשתמש ברשת נוירונים על מנת לזהות את הקשר בין הקלט שלנו התשובות שהבן אדם עונה לבין הפלט שלנו מה הבן אדם רוצה לאכול.

### Socket

מודול המעניק ממשק Low-Level ליצירת תקשורת ברשת בין שני מכשירים.

### Google Colab

מקום קוד פתוח המאפשר לבנות מכונה לומדת בצורה נוחה, עם אפשרות להרצה על השרתים של גוגל, דבר זה מאוד שימושי מכיוון שלא כל מחשב יכול לאמן מכונה וזה יכול לקחת ימים. לכן, Google Colab זה מקום טוב המאפשר אימון מכונה מהיר הרבה יותר על שרת עם כוח עיבוד מטורף.

### OpenCV

חבילת תוכנה המעניקה כלי לזיהוי אובייקטים בתמונה וכלים שונים לעיבוד תמונה.

### Auto-py-to-exe

חבילת תוכנה המאפשרת קימפול של קובץ Python.

### TensorFlow

TensorFlow היא ספריית קוד פתוח ללמידת מכונה, המפותחת על ידי חברת גוגל לבנייה ואימון רשתות ניורונים.

## טיפול בחריגות

בצד השרת ובצד הלקוח יש טיפולים שונים בחריגות.

### טיפול בחריגות בצד השרת:

* הריגת תהליכים המאזינים בפורט השרת – כאשר השרת מנסה להעלות והוא לא מצליח בגלל שגיאת מערכת, זה אומר שמישהו מאזין בפורט שהשרת. לכן המערכת תהרוג אותו והשרת ינסה להעלות שוב.
* במקרה שהלקוח מתנתק נפסיק לשלוח לו הודעות והשרת לא יקרוס.

### טיפול בחריגות בצד הלקוח:

* בתחילת התהליך המשתמש מתבקש להכניס IP של השרת. אם השרת לא פעיל תופיע שגיאת שרת.
* במקרה בו השרת נופל או ללקוח אין אינטרנט גם אז הלקוח יועבר למסך שגיאה.

## Open sources

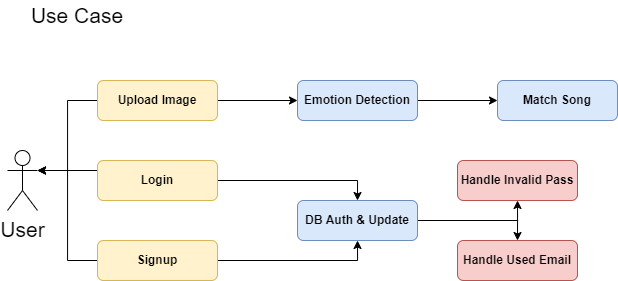
<https://flutter.dev/>

<https://pub.dev/>

במקור הראשון מצויים קטעי קוד בflutter שעזרו לי לפתוח. במקור השני יש תוספים שניתן להוסיף לאפליקציה שעוזרים לפתח קוד, כמו תוסף שנותן זיהוי גודל מסך כך שניתן להפוך את המסך לריספונסיבי.

<https://stackoverflow.com/>.

בנוסף לכך, ניהלתי דיון באתר StackOverflow על מנת לפתור bug באפליקציה שלא היה לו כל תיעוד בגוגל.

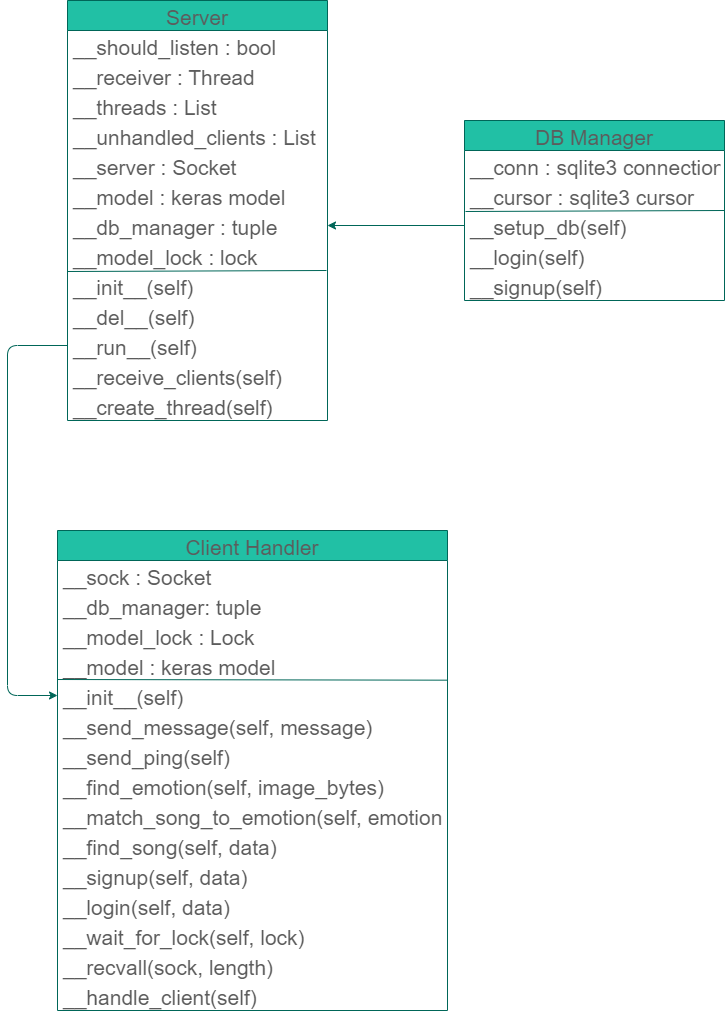


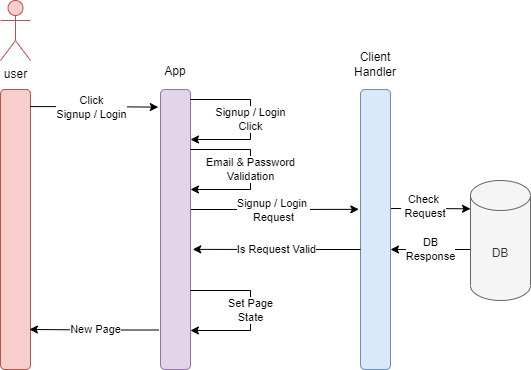
## Use Case

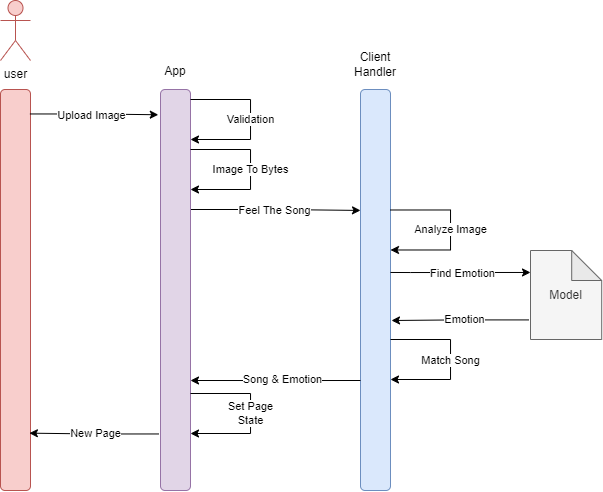
## מדריך משתמש

## Screen Flow Diagram

## Server UML



Sequence Diagrams



## בסיס הנתונים

בסיס הנתונים בפרויקט הגמר שלי אחראי על שמירת המידע על המשתמשים לצורך הרשמה והתחברות.

לכל משתמש יש 2 רשומות:

email - כתובת המייל של המשתמש איתה הוא נרשם לאפליקציה, נשמרת כטקסט. האימייל הוא המפתח הראשי, ולכן יכול להיות רק אימייל אחד לכל לקוח ( הגיוני סך הכל לא? ).

password - הסיסמה של המשתמש איתה הוא נרשם לאפליקציה, נשמרת כטקסט.

סכמה



בסיס הנתונים שלי הוא בסיס נתונים מסוג SQLITE. הטבלה נשמרת או נוצרת על המחשב שמריץ את הקובץ של השרת.

## חולשות מטופלות

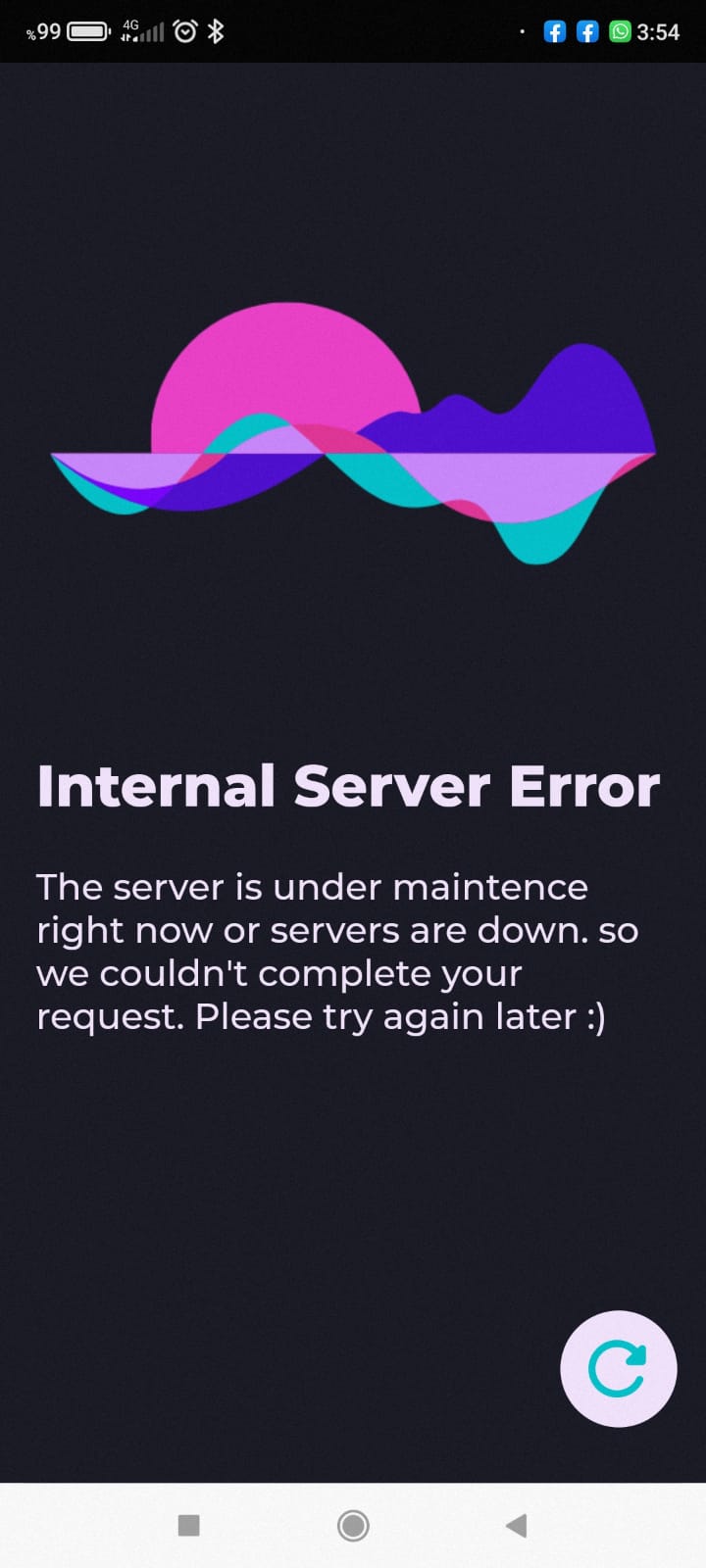
* Dos Attack – לשרת מגבלת טיפול, כלומר הוא יכול לטפל בכמות מסוימת של אנשים במקביל, כל שאר הבקשות יוכנסו לQueue. לכן מתקפה זו לא אפקטיבית, היא תאט את פעילותו אך לא תגרום לו לקרוס.
* SQL injection – השרת פועל מול מסד נתונים מסוג SQLITE לכן הוא חשוף להזרקות קוד, בעיה זו תוקנה על ידי בדיקת התווים לפני שליחתם לשרת ובדיקתם מול SQL.
* התחברות בתור מישהו אחר – השרת מבצע אימות משתמשים לכן משתמש יכול להיכנס רק עם המשתמש והסיסמא שבחר בכניסה.

## תמונה שמכילה טקסט, כרטיס ביקור התיאור נוצר באופן אוטומטימסכים

### מסך ספלאש

תפקיד מסך זה הוא להראות אנימציה טעינה נחמדה בזמן שהאפליקציה מתחברת לשרת, אם קיים קובץ עם שם משתמש וסיסמא האפליקציה תתחבר אוטומטית ללא צורך במעבר לדף התחברות או הרשמה.

תיבת טקסט Server IP – תפקידה לקבל IP של השרת מהמשתמש. אופציה זאת קיימת מכיוון שעל מנת שהשרת ירוץ מחוץ לLocalHost הוא צריך לקבל IP, אך הסרבר לא יושב על מחשב קבוע לכן צריך לבקש את ה IP של הסרבר.



### מסך שגיאת התחברות

תפקיד מסך זה הוא להודיע למשתמש שקרתה שגיאה בעת ההתחברות לשרת, השגיאה יכולה להיות בIP שהמשתמש הזין ויכולה להיות בגלל שהשרת כרגע לא פעיל.

כפתור Try Again – הכפתור נועד כדי לאפשר למשתמש חזרה נוחה למסך הטעינה שראינו קודם לכן.

### תמונה שמכילה טקסט, אלקטרוניקה התיאור נוצר באופן אוטומטימסך התחברות

תפקיד מסך זה הוא לאפשר למשתמש קיים במערכת, לבצע התחברות אל האפליקציה.

תיבת טקסט Email - תפקידה לאפשר למשתמש קיים להכניס את כתובת המייל איתה הוא רשום למערכת.

תיבת טקסט Password - תפקידה לאפשר למשתמש קיים להכניס את הסיסמה איתה הוא רשום למערכת. הסיסמה מוסתרת, ואפשר לחשוף / להסתיר אותה מחדש ע"י לחיצה על כפתור העין.

כפתור Login – תפקידו להעביר את הפרטים שהוזנו בתיבות הטקסט לשרת, ולבדוק האם הקלט בתיבות הטקסט נכון ותקין. אם המשתמש קיים, הוא מועבר למסך הראשי.

כפתור Don’t have an account? Sign up – תפקידו להעביר את המשתמש מהמסך הנוכחי למסך ההרשמה.

• במידה והמשתמש אינו קיים במערכת, תוצג הודעת שגיאה.

### מסך הרשמה

תפקיד מסך זה הוא לאפשר למשתמש לא קיים במערכת, לבצע הרשמה אל האפליקציה.

תיבת טקסט Email - תפקידה לאפשר למשתמש קיים להכניס את כתובת המייל איתה הוא רשום למערכת.

תיבת טקסט Password - תפקידה לאפשר למשתמש קיים להכניס את הסיסמה איתה הוא רשום למערכת. הסיסמה מוסתרת, ואפשר לחשוף / להסתיר אותה מחדש ע"י לחיצה על כפתור העין.

כפתור Signup – תפקידו להעביר את הפרטים שהוזנו בתיבות הטקסט לשרת, ולבדוק האם הקלט בתיבות הטקסט נכון ותקין. אם המשתמש לא קיים, הוא מועבר למסך הראשי.

כפתור Already have an account? Login – תפקידו להעביר את המשתמש מהמסך הנוכחי למסך ההתחברות.

• במידה והמשתמש קיים במערכת, תוצג הודעת שגיאה.

### מסך בית

תפקיד מסך זה הוא לאפשר למשתמש להעלות תמונה אל השרת, ברגע שהמשתמש מעלה תמונה הוא לוחץ על כפתור ה Feel Your Music. ומשם יעבור למסך שיראה לו את הרגש והשיר שזוהה.

כפתור הפנים – תפקיד כפתור זה הוא לאפשר למשתמש להיכנס לגלריה מתוך האפליקציה ולבחור תמונה.

כפתור ה Feel Your Music – תפקיד כפתור זה הוא לאפשר מעבר למסך הבא, ולשלוח את התמונה לשרת.

כפתור הבית – כפתור הבית יביא את המשתמש למסך הבית, המסך הנוכחי.

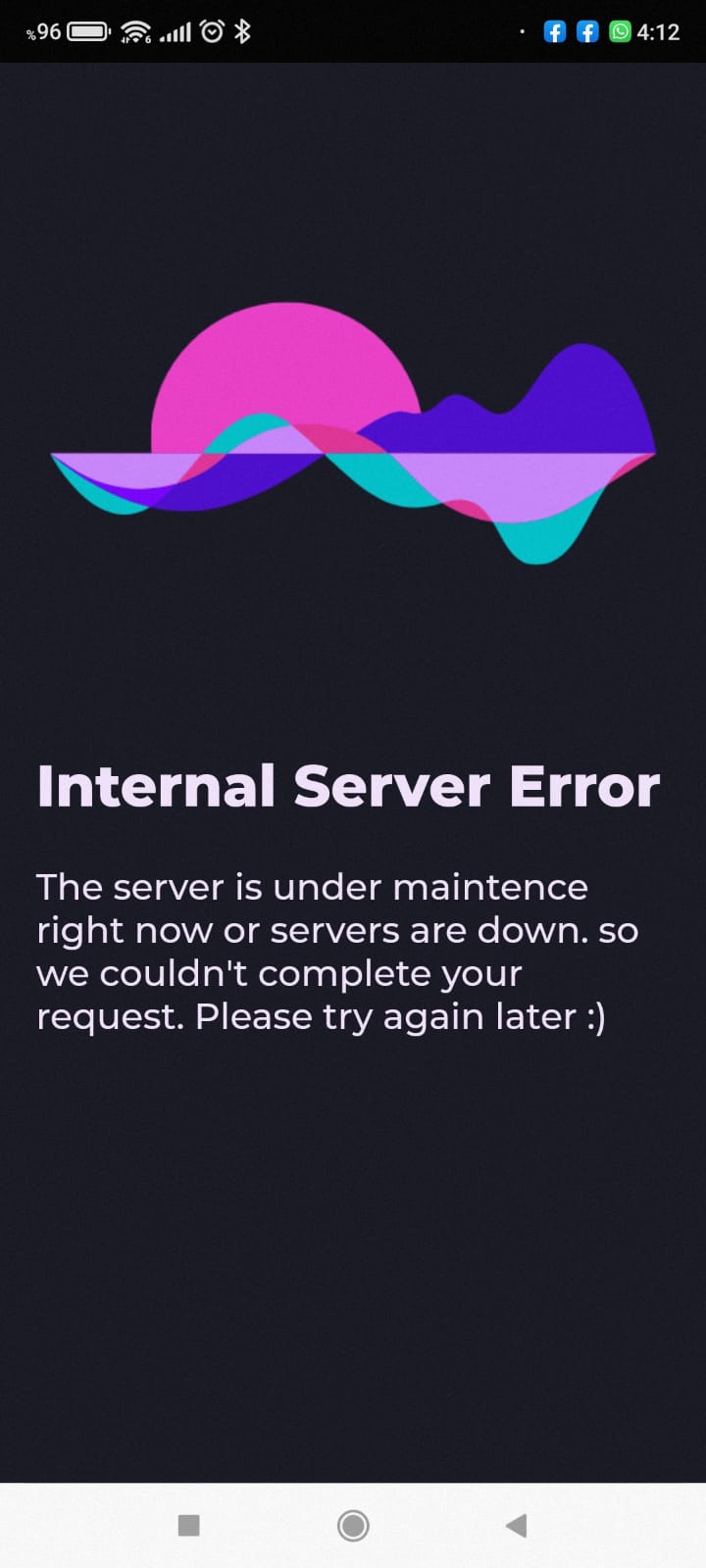
לאחר שמשתמש יעלה את התמונה שלו התמונה תחליף את תמונת הפנים שראינו קודם.

• במידה ואין תמונה תיזרק שגיאה למשתמש שעליו לבחור אחת.

### מסך טעינה

עד שזיהוי התמונה ישלח מהאפליקציה אל השרת ומהשרת חזרה אל האפליקציה, תוצג אנימציית טעינה אותה ראינו בהתחלה.

תפקיד מסך זה הוא להמיר את התמונה לביטים ולשלוח אותם אל השרת, ובנוסף לשעשע את המשתמש בזמן ההמתנה.



### מסך שגיאת שרת

תפקיד מסך זה הוא להודיע למשתמש על שגיאה שקרתה אצל השרת, אם הוא נפל או שקרתה בעיה.

אין דרך חזרה ממסך זה על המשתמש לסגור את האפליקציה.

### מסך תוצאה

לאחר שהתמונה הגיע מהשרת היא תועבר ממסך הטעינה למסך הזה. תפקיד המסך הוא להציג את תוצאות השרת ולאפשר מעבר ל YouTube לשמוע את השיר.

כפתור Go & Listen On Youtube – מאפשר למשתמש מעבר מהאפליקציה אל YouTube על מנת לשמוע את השיר שהותאם לו.

כפתור הבית – כפתור זה מאפשר למשתמש לחזור אל מסך הבית לזיהוי חוזר של שירים.

## מדריך למפתח

### main.py

בקוד הראשי יש לנו הרצה ראשונית של השרת, אם השרת נפל בגלל שגיאת חיבור ל Socket הסקריפט יחסל את התהליך המאזין בפורט 2508 ויריץ את השרת שוב כך שיוכל להקשיב לפורט 2508.

def handle\_os\_error():

    result = subprocess.check\_output("netstat -ano|findstr 2508", shell=True).decode()

    result = result[71:].strip()

    os.system(f"tskill {result}")

### server.py

קובץ זה אחראי על הפעלת השרת, הקובץ מתחיל להאזין בפורט 2508 וה איי פי של השרת יקבע בקובץ זה לפי כתובת האיי פי של המחשב. אם השרת כבר עסוק הוא יכניס את המשתמשים לתור, אם הוא פנוי הוא יוציא את המשתמש הראשון שהגיע מהתור.

קבועים

    \_\_SERVER\_INFO = (socket.gethostbyname(socket.gethostname()), 2508)

    \_\_MAX\_CLIENTS = 2

משתנים

self.\_\_should\_listen = True

            self.\_\_receiver = Thread(target=self.\_\_receive\_clients, args=[])

            self.\_\_threads = []

            self.\_\_unhandled\_clients = []

            self.\_\_server = server

            self.\_\_model = keras.models.load\_model("model")

            self.\_\_db\_manager = (DBManger(), Lock())

            self.\_\_model\_lock = Lock()

פונקציות

\_\_init\_\_ - הפונקציה מאתחלת את המשתנים הלוקאלים של השרת ומריצה את הפונקציה run.

\_\_del\_\_ - הפונקציה משנה את מצב אחד המשתנים על מנת לגרום לשרת להבין שהוא צריך להפסיק להקשיב ולקבל לקוחות, כלומר לסגור את הסוקט.

\_\_run – אם השרת לא עסוק הפונקציה תוציא לקוח חדש מהתור ותטפל בו, אם הוא עסוק היא תחכה עד שהוא יסיים את ענייניו עם הלקוחות האחרים.

\_\_receive\_clients – הפונקציה אחראית על הכנסת לקוחות אל תור מסויים הפונקציה פועלת כתהליך נפרד בתוכנית על מנת שהדברים יעשו במקביל.

\_\_create\_thread – הפונקציה יוצרת תהליך לטיפול בלקוח עם המשתנים המתאימים.

  def \_\_run(self) -> None:

        self.\_\_receiver.start()

        try:

            while True:

                self.\_\_threads = [t for t in self.\_\_threads if t.is\_alive()]

                if len(self.\_\_threads) <= self.\_\_MAX\_CLIENTS and len(self.\_\_unhandled\_clients) != 0:

                    print("handling a new client so exciting ^-^")

                    self.\_\_create\_thread(self.\_\_unhandled\_clients.pop(0))

        except KeyboardInterrupt:

            pass

### db\_manger.py

משתנים

        self.\_\_conn = sqlite3.connect("clients.db", check\_same\_thread=False)

        self.\_\_cursor = self.\_\_conn.cursor(

פונקציות

\_\_init\_\_ - הפונקציה מאתחלת את המשתנים הלוקאלים, ומריצה את הפונקציה setup\_db.

\_\_setup\_db – הפונקציה יוצרת מסד נתונים אם הוא לא קיים.

\_\_login – הפונקציה בודקת אם אפשר לחבר משתמש לאפליקציה, אם האימייל והסיסמא קיימים במסד הנתונים היא תחזיר true אחרת היא תחזיר false.

\_\_signup - הפונקציה בודקת אם אפשר לרשום משתמש חדש למסד הנתונים, אם האימייל לא קיים היא תחזיר true אחרת היא תחזיר false.

### Client\_handler.py

משתנים

        self.\_\_sock = sock

        self.\_\_db\_manager = db\_manager

        self.\_\_model\_lock = model\_lock

        self.\_\_model = model

פונקציות

\_\_init\_\_ - הפונקציה מאתחלת את המשתנים הלוקאלים, ומריצה את הפונקציה handle\_client.

\_\_send\_message – הפונקציה שולחת הודעה לפי הפורמט שנקבע בצורה בטוחה עם try ו except.

\_\_send\_ping – הפונקציה בודקת אם הסוקט עדיין בחיים והמשתמש לא התנתק.

\_\_find\_emotion – הפונקציה מריצה את המודל ומזהה את הרגש בפנים של התמונה שקיבלה.

\_\_match\_song\_to\_emotion – הפונקציה משתמשת במילון, היא מקבלת רגש ומתאימה לו שיר לפי המילון.

\_\_find\_song – הפונקציה מחכה למנעול מוצאת את הרגש מתאימה לו שיר ומחזירה את התשובה.

\_\_signup – הפונקציה מבצעת פעולת הירשמות ובודקת האם זה אפשרי תחזיר true אם אפשר ו false אם לא.

\_\_login – הפונקציה מבצעת פעולת התחברות ובודקת האם זה אפשרי תחזיר true אם אפשר ו false אם לא.

\_\_wait\_for\_lock – הפונקציה מחכה למנעול מסוים שישתחרר ופונקציה אחרת תוכל לגשת אליו.

\_\_recvall – הפונקציה מקבלת הודעה גדולה עד תומה.

\_\_handle\_client - הפונקציה מקבלת הודעות מהמשתמש ומפנה אותו אל הפונקציה הנכונה, כל חצי דקה הפונקציה תשלח הודעה לבדוק אם המשתמש עדין קיים ולא קרה כלום לסוקט.

    def \_\_handle\_client(self) -> None:

        options = {1: self.\_\_login, 2: self.\_\_signup, 3: self.\_\_find\_song}

        timeout = 30

        timeout\_start = time.time()

        try:

            while True:

                packet = self.\_\_sock.recv(5)

                if packet:

                    try:

                        # extracting code and length

                        code = int(packet[0])

                        if code != 4:

                            length = int.from\_bytes(packet[1:5], byteorder="little")

                            # receive data using the length we got

                            data = self.\_\_recvall(self.\_\_sock, length)

                            options[code](data)

                        else:

                            raise ConnectionResetError

                    except KeyError:

                        pass

                    except Exception as e:

                        self.\_\_send\_message("3")

                        print(str(e))

                # every 30 seconds check if client still alive

                if time.time() >= timeout\_start + timeout:

                    if not self.\_\_send\_ping():

                        raise ConnectionResetError

                    timeout\_start = time.time()

        except ConnectionResetError:

            print("oh the clinet left boo hoo :(")

        except ConnectionAbortedError:

            print("oh the clinet left boo hoo :(")

        except KeyboardInterrupt:

            pass

    def \_\_find\_emotion(self, image\_bytes: bytearray) -> str:

        start = time.time()

        print("image received lets find some emotion!\n")

        # bytes array to image

        image = Image.open(io.BytesIO(image\_bytes))

        test\_image = cv2.cvtColor(np.array(image), cv2.COLOR\_RGB2BGR)

        gray\_image = cv2.cvtColor(test\_image, cv2.COLOR\_BGR2GRAY)

        face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")

        faces = face\_cascade.detectMultiScale(gray\_image, 1.1, 4)

        # detect emotion

        emotions = []

        for x, y, w, h in faces:

            cv2.rectangle(test\_image, (x, y), (x + w, y + h), (255, 0, 0))

            roi\_gray = gray\_image[y : y + h, x : x + h]

            roi\_gray = cv2.resize(roi\_gray, (48, 48))

            image\_pixels = img\_to\_array(roi\_gray)

            image\_pixels = np.expand\_dims(image\_pixels, axis=0)

            image\_pixels /= 255

            predictions = self.\_\_model.predict(image\_pixels)

            max\_index = np.argmax(predictions[0])

            emotion\_detection = ["Angry", "Disgust", "Fear", "Happy", "Sad", "Surprise", "Neutral"]

            emotion\_prediction = emotion\_detection[max\_index]

            emotions.append(emotion\_prediction)

            break

        # if there is more than one face we will take the first emotion

        if len(emotions) > 0:

            return emotions[0]

        return "Neutral"

### מסך התחברות והירשמות

  bool \_isEmailValid(String email) {

    return RegExp(

            r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'\*+-/=?^\_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+")

        .hasMatch(email);

  }

  bool \_isPasswordValid(String password) {

    return RegExp(r'[^A-Za-z0-9]+').hasMatch(password);

  }

### Helper.dart

קובץ הנועד לעזור לקרוא ולכתוב לקבצים

Future<String> get configDirPath async {

  var directory = await getApplicationDocumentsDirectory();

  return directory.path;

}

Future<void> createConfig() async {

  var directory = Directory(configDirPath.toString());

  directory.create(recursive: true);

}

Future<String> get configFilePath async {

  return '${await configDirPath}/config.bin';

}

Future<File> get configFile async {

  String path = await configFilePath;

  return File(path);

}

Future<void> writeConfig(String content) async {

  final file = await configFile;

  file.writeAsBytes(utf8.encode(content));

}

### login\_signup.dart

קובץ הנועד להכיל את ה widgets המשותפים ל login ול signup על מנת למנוע חזרה על קוד, ולהקל על קריאת הקוד.

### menu.dart

הקובץ נועד להכיל את ה widgets של menu על מנת להקל בקריאת הקוד.

### socket.dart

קובץ המכיל class המרכז את כל הפונקציות ההכרחיות של socket שצריך כדי להתחבר לשלוח הודעות ולקבל הודעות בצורה נוחה.

  Future<bool> connectServer(BuildContext context) async {

    try {

      await \_displayTextInputDialog(context);

      socket = await Socket.connect(\_textFieldController.text, 2508,

          timeout: const Duration(seconds: 4));

      stream = socket.listen(

          (data) {

            if (data.contains(49)) {

              completer.complete(extractData(data));

              stream.pause();

            }

          },

          onDone: () {},

          cancelOnError: true,

          onError: (error) {

            completer.complete("error");

          });

      if (!stream.isPaused) {

        stream.pause();

      }

      return true;

    } on SocketException {

      await Navigator.pushNamed(

        context,

        '/connectionE',

      );

      return false;

    }

  }

  Future<dynamic> receiveMessage() {

    stream.resume();

    return completer.future;

  }

  Future<String> sendAndWait(

      BuildContext context, String message, int code) async {

    try {

      completer = Completer();

      sendMessage(message, code);

      String reply = await receiveMessage();

      if (reply == "error" || reply == "3") {

        MSocket.pushReplacement(context, "/serverDownE", this);

        return "2";

      }

      return reply;

    } catch (erorr) {

      MSocket.pushReplacement(context, "/serverDownE", this);

    }

    return "2";

  }

### קוד המכונה

חלק מקוד המכונה האחראי על הוספת את השכבות הנסתרות בין המכונה ואימון המכונה.

# 1st convolution layer

model.add(Conv2D(64, kernel\_size=(3,3), activation='relu', input\_shape=(x\_train.shape[1:])))

model.add(Conv2D(64, kernel\_size=(3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))

model.add(Dropout(0.5))

# 2nd convolution layer

model.add(Conv2D(64, (3,3), activation='relu'))

model.add(Conv2D(64, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))

model.add(Dropout(0.5))

# 3rd convolution layer

model.add(Conv2D(128, (3,3), activation='relu'))

model.add(Conv2D(128, (3,3), activation='relu'))

model.add(BatchNormalization())

model.add(MaxPooling2D(pool\_size=(2,2), strides=(2,2)))

model.add(Flatten())

# fully connected neural networks

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(num\_labels, activation='softmax'))

model.summary()

# compiling the model

model.compile(loss=categorical\_crossentropy,

             optimizer='sgd',

             metrics=['accuracy'])

# training the model

model.fit(x\_train, y\_train,

          batch\_size=batch\_size,

          epochs=epochs,

          verbose=1,

          validation\_data=(x\_test, y\_test),

          shuffle=True)

model.save('facial\_regcognition\_model')

## רפלקציה

### חוויה כללית מתהליך העבודה

תהליך העבודה היה מהנה מכיוון שחקרתי ולמדתי דברים חדשים, אמנם הייתי צריך לקודד מלא, אבל היה כיף. ובסופו של דבר יצאתי עם פרויקט שאני יכול להתגאות בו.

### מה קיבלתי מהפרויקט

פרויקט הגמר הרחיב את הידע שלי ועזר לי ללמוד שני תחומים מאוד מעניינים, תחום הלומדה ואת נושא ה GUI. הפרויקט ישמש אותי להמשך כמשהו שאוכל להציג בראיונות ובפנים אנשים שונים. קיבלתי כלים שונים כמו מסוגלות, התמדה למרות שקשה והמון ידע חדש.

### כלים שאקח להמשך הדרך

אקח איתי להמשך את הידע הנרחב בטכנולוגיות השונות שלמדתי במהלך הפרויקט, ואת היכולת לנהל זמן באופן נכון.

### אתגרים

האתגר המרכזי אשר עמד בפניי הוא הזמן, בתור תלמיד י"ב שרוב בגרויותיו נדחו לשנת י"ב היה קשוח לעבוד במקביל לבגרויות ולתוכנית מגשימים. הדבר העיקרי אשר עזר לי להתגבר על קושי זה היה תכנון זמן נכון, עמידה בזמנים, ומשמעת עצמית בכל מה שקשור למטרות ויעדים בפרויקט.

### מסקנות

מסקנותיי מהפרויקט הזה הן שבאינטרנט יש הכל ואם קצת רצון וקצת יכולת אפשר ללמוד באמת הכל, ומשם השמיים הם הגבול. בנוסף לכך, גיליתי שאני נהנה לכתוב GUI ולעצב דברים שונים.

### מה הייתי עושה אחרת לו הייתי מתחיל היום

לא הייתי עושה כלום אחרת, אני מאוד מרוצה וגאה בעבודה שלי ושמח שבחרתי בה. אני למדתי את הנושאים שרציתי, עמדתי בזמנים, והייתה חוויה יוצאת דופן, גם אם היה קשה בדרך לומדים מזה ולכן לא הייתי משנה כלום.

### מה היה קורה אחרת אם העבודה הייתה יותר יעילה עבורי

העבודה הייתה יעילה עבורי, עבדתי בשיטה נוחה, ניהלתי את הזמן נכון ולכן לא הייתה שום בעיית יעילות עבורי.

### רעיון לפיתוח עתידי

לאחר סיום הפיתוח ניתן יהיה להוסיף פיצ'רים לאפליקציה ולהפוך אותה לרשת חברתית. פיצ'רים כמו היסטוריית שיחות, הוספת תמונה, שיתוף התמונה + השיר שיצא. שילוב צ'אט בין היוזרים השונים. לאחר שהאפליקציה תהיה גמורה אשחרר גרסה ראשונית לחנות האפליקציות, ולאחר מכן אהפוך אותה לרשת חברתית עם הפיצ'רים שהוזכרו ואשחרר עדכון לתוכנה.

## ביבליוגרפיה

Flutter:

* <https://flutter.dev/>
* <https://youtu.be/1ukSR1GRtMU>
* <https://pub.dev/>

Machine learning:

* <https://www.coursera.org/learn/machine-learning>

SQL:

* <https://www.w3schools.com/sql/>

OpenCV:

* <https://opencv.org/>
* <https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>
* <https://docs.opencv.org/3.4/da/d60/tutorial_face_main.html>

NumPy:

* <https://numpy.org/>

socket:

* <https://docs.python.org/3/library/socket.html>

auto-py-to-exe:

* <https://pypi.org/project/auto-py-to-exe/>

keras:

* <https://keras.io/>

TensorFlow:

* <https://www.tensorflow.org/>

Goggle Colab:

* <https://colab.research.google.com/?utm_source=scs-index>