

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](#_Toc104436757)

[תיאור כללי 6](#_Toc104436758)

[תהליך המחקר 6](#_Toc104436759)

[הסיבה לבחירת הפרויקט 6](#_Toc104436760)

[הבעיה עימה התמודדתי 6](#_Toc104436761)

[סוגי משתמשים / קהל היעד 7](#_Toc104436762)

[יעדים 7](#_Toc104436763)

[צד שרת המספק 7](#_Toc104436764)

[צד לקוח המספק 7](#_Toc104436765)

[המוטיבציה לעבודה 7](#_Toc104436766)

[הצורך שעליו עונה הפרויקט והפתרון שהפרויקט בא לתת 7](#_Toc104436767)

[אילוצים ודרישות 8](#_Toc104436768)

[אילוצי זמן 8](#_Toc104436769)

[אילוצי תוכנה וידע 8](#_Toc104436770)

[אילוצי למידה 8](#_Toc104436771)

[פתרונות קיימים 8](#_Toc104436772)

[המלצת שירים לפי רגשות או טעם 8](#_Toc104436773)

[זיהוי רגש בפנים 8](#_Toc104436774)

[דרישות מערכת 8](#_Toc104436775)

[שרת 8](#_Toc104436776)

[אפליקציה 8](#_Toc104436777)

[פיצ'רים ותהליכים עיקריים 9](#_Toc104436778)

[ציר טכנולוגי 9](#_Toc104436779)

[ארכיטקטורת הפרויקט 10](#_Toc104436780)

[הצגת הפתרון המוצע 10](#_Toc104436781)

[רכיבים 11](#_Toc104436782)

[Face Recognition 11](#_Toc104436783)

[Server 11](#_Toc104436784)

[DB (מסד נתונים) 11](#_Toc104436785)

[App 11](#_Toc104436786)

[תיאור זרימת המידע במערכת 12](#_Toc104436787)

[מבני הנתונים בפרויקט 12](#_Toc104436788)

[רשימה - List 12](#_Toc104436789)

[מערך בתים – bytearray 12](#_Toc104436790)

[מילון - Dictionary 12](#_Toc104436791)

[טאפל - Tuple 12](#_Toc104436792)

[ארכיטקטורת רשת 13](#_Toc104436793)

[תיאור פרוטוקולי תקשורת 13](#_Toc104436794)

[פירוט על הטכנולוגיות 14](#_Toc104436795)

[Python 14](#_Toc104436796)

[NumPy/Keras 14](#_Toc104436797)

[Flutter 14](#_Toc104436798)

[SQLITE 14](#_Toc104436799)

[Neural network 14](#_Toc104436800)

[Socket 14](#_Toc104436801)

[Google Colab 14](#_Toc104436802)

[OpenCV 14](#_Toc104436803)

[Auto-py-to-exe 14](#_Toc104436804)

[Use Case 15](file:///C:\Users\Gal%20Pahima\Desktop\בית%20ספר.docx#_Toc104436805)

[מדריך משתמש 15](#_Toc104436806)

[Screen Flow Diagram 15](file:///C:\Users\Gal%20Pahima\Desktop\בית%20ספר.docx#_Toc104436807)

[Server UML 16](#_Toc104436808)

[בסיס הנתונים 17](#_Toc104436809)

[חולשות מטופלות 17](#_Toc104436810)

[מסכים 18](#_Toc104436811)

[מסך ספלאש 18](#_Toc104436812)

[מסך שגיאת התחברות 18](file:///C:\Users\Gal%20Pahima\Desktop\בית%20ספר.docx#_Toc104436813)

[מסך התחברות 19](#_Toc104436814)

[מסך הרשמה 19](#_Toc104436815)

[מסך בית 20](#_Toc104436816)

[מסך טעינה 21](#_Toc104436817)

[מסך שגיאת שרת 21](#_Toc104436818)

[מסך תוצאה 22](#_Toc104436819)

[מדריך למפתח 23](#_Toc104436820)

[main.py 23](#_Toc104436821)

[server.py 24](#_Toc104436822)

[db\_manger.py 27](#_Toc104436823)

[Client\_handler.py 29](#_Toc104436824)

[מסך ספלאש 35](#_Toc104436825)

[מסך שגיאת התחברות 38](#_Toc104436826)

[מסך התחברות 40](#_Toc104436827)

[מסך הירשמות 46](#_Toc104436828)

[מסך בית 52](#_Toc104436829)

[מסך טעינה 56](#_Toc104436830)

[מסך שגיאת שרת 59](#_Toc104436831)

[מסך תוצאה 61](#_Toc104436832)

[Helper.dart 66](#_Toc104436833)

[login\_signup.dart 67](#_Toc104436834)

[menu.dart 74](#_Toc104436835)

[socket.dart 80](#_Toc104436836)

[קוד המכונה 85](#_Toc104436837)

[רפלקציה 88](#_Toc104436838)

[חוויה כללית מתהליך העבודה 88](#_Toc104436839)

[מה קיבלתי מהפרויקט 88](#_Toc104436840)

[כלים שאקח להמשך הדרך 88](#_Toc104436841)

[אתגרים 88](#_Toc104436842)

[מסקנות 88](#_Toc104436843)

[מה הייתי עושה אחרת לו הייתי מתחיל היום 88](#_Toc104436844)

[מה היה קורה אחרת אם העבודה הייתה יותר יעילה עבורי 88](#_Toc104436845)

[ביבליוגרפיה 89](#_Toc104436846)

# 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.

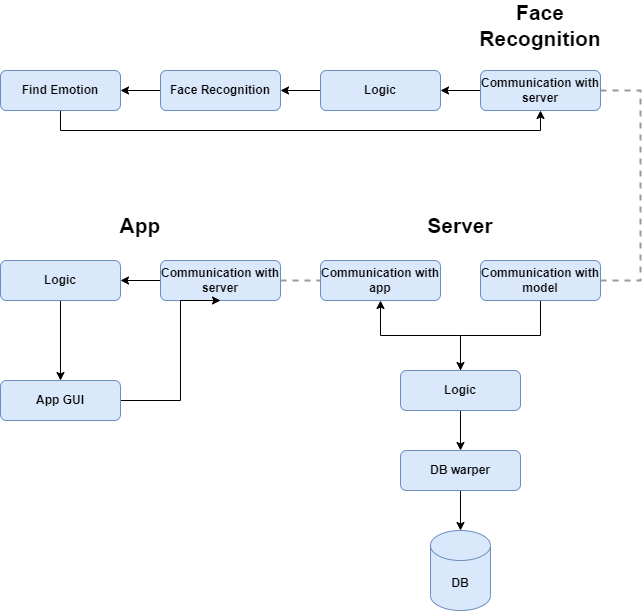
## ציר טכנולוגי

* 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).

# 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.

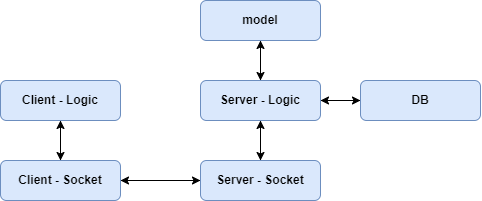
### DB (מסד נתונים)

* 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

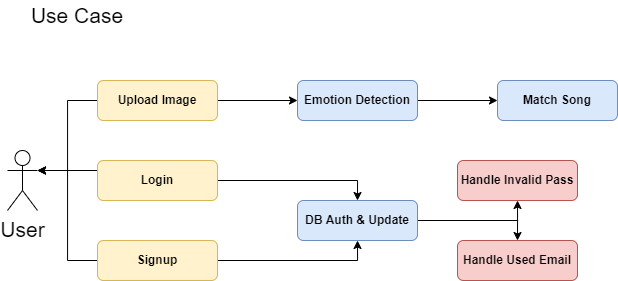
חבילת תוכנה המעניקה כלי לזיהוי אובייקטים בתמונה וכלים שונים לעיבוד תמונה.

### 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.

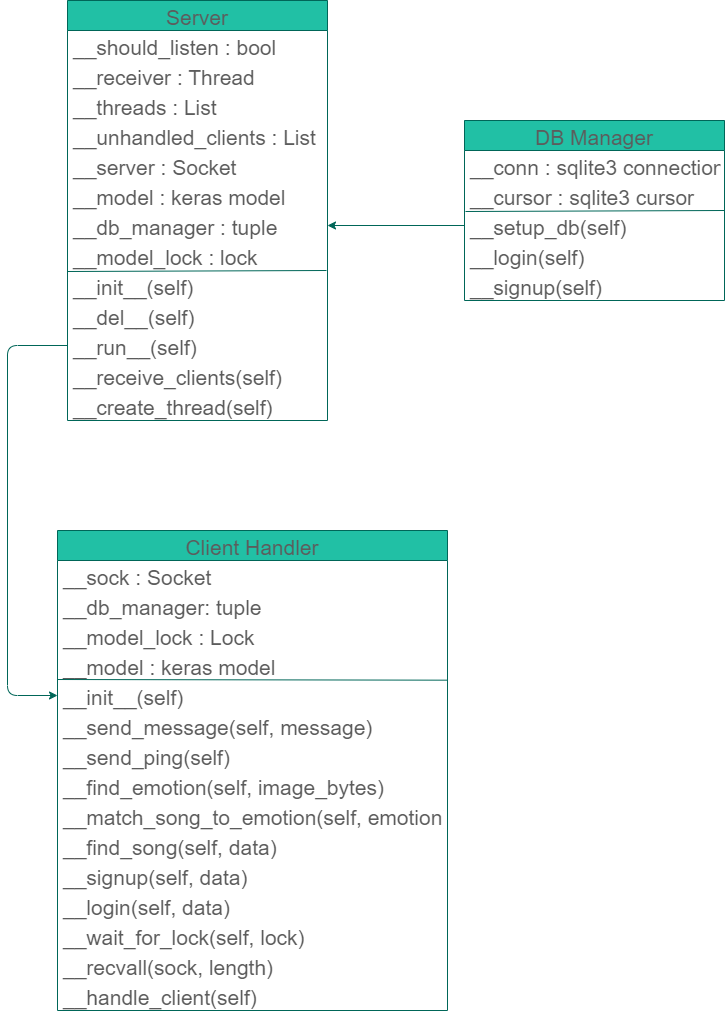


## Use Case

## מדריך משתמש

## Screen Flow Diagram

## Server UML



## 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

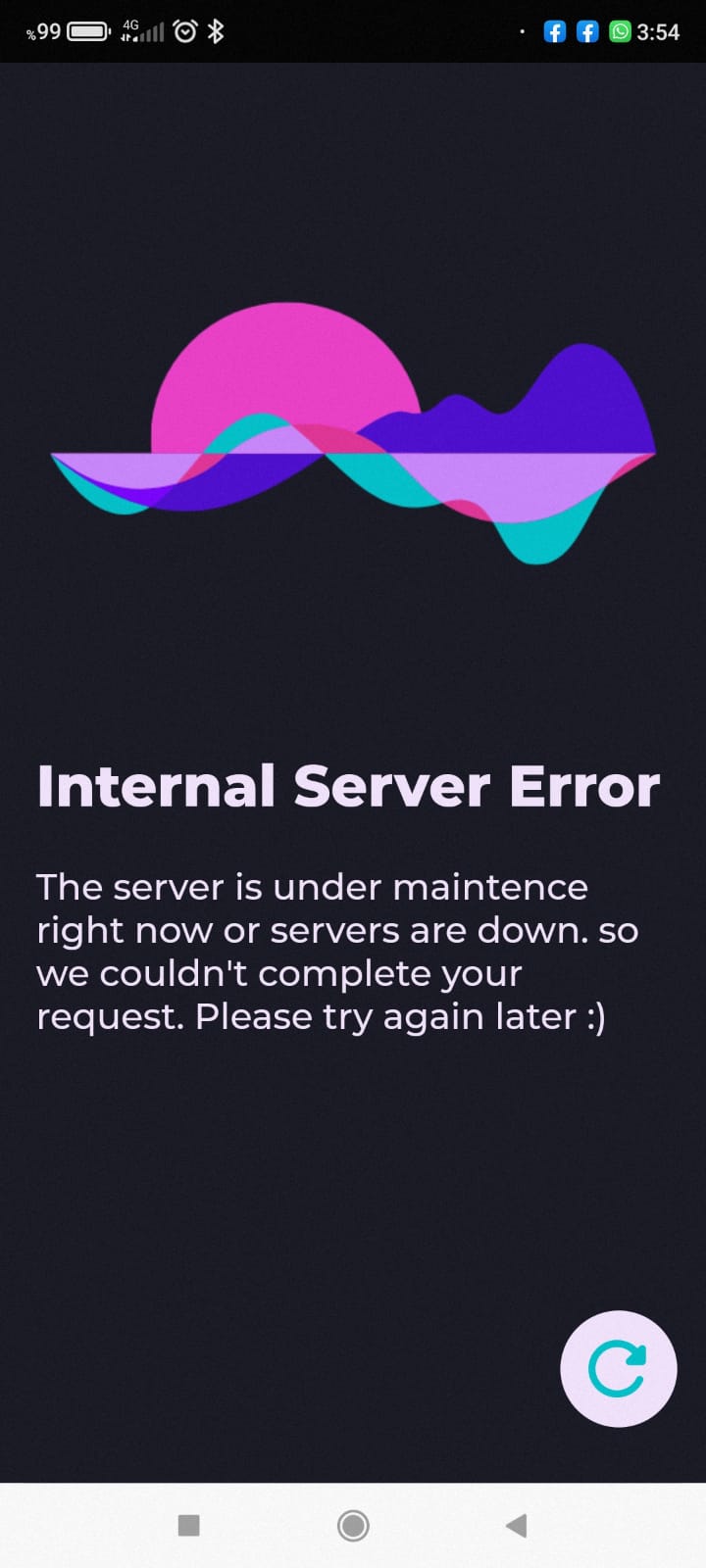
* Dos Attack – לשרת מגבלת טיפול, כלומר הוא יכול לטפל בכמות מסוימת של אנשים במקביל, כל שאר הבקשות יוכנסו לQueue. לכן מתקפה זו לא אפקטיבית, היא תאט את פעילותו אך לא תגרום לו לקרוס.
* SQL injection – השרת פועל מול מסד נתונים מסוג SQLITE לכן הוא חשוף להזרקות קוד, בעיה זו תוקנה על ידי בדיקת התווים לפני שליחתם 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.

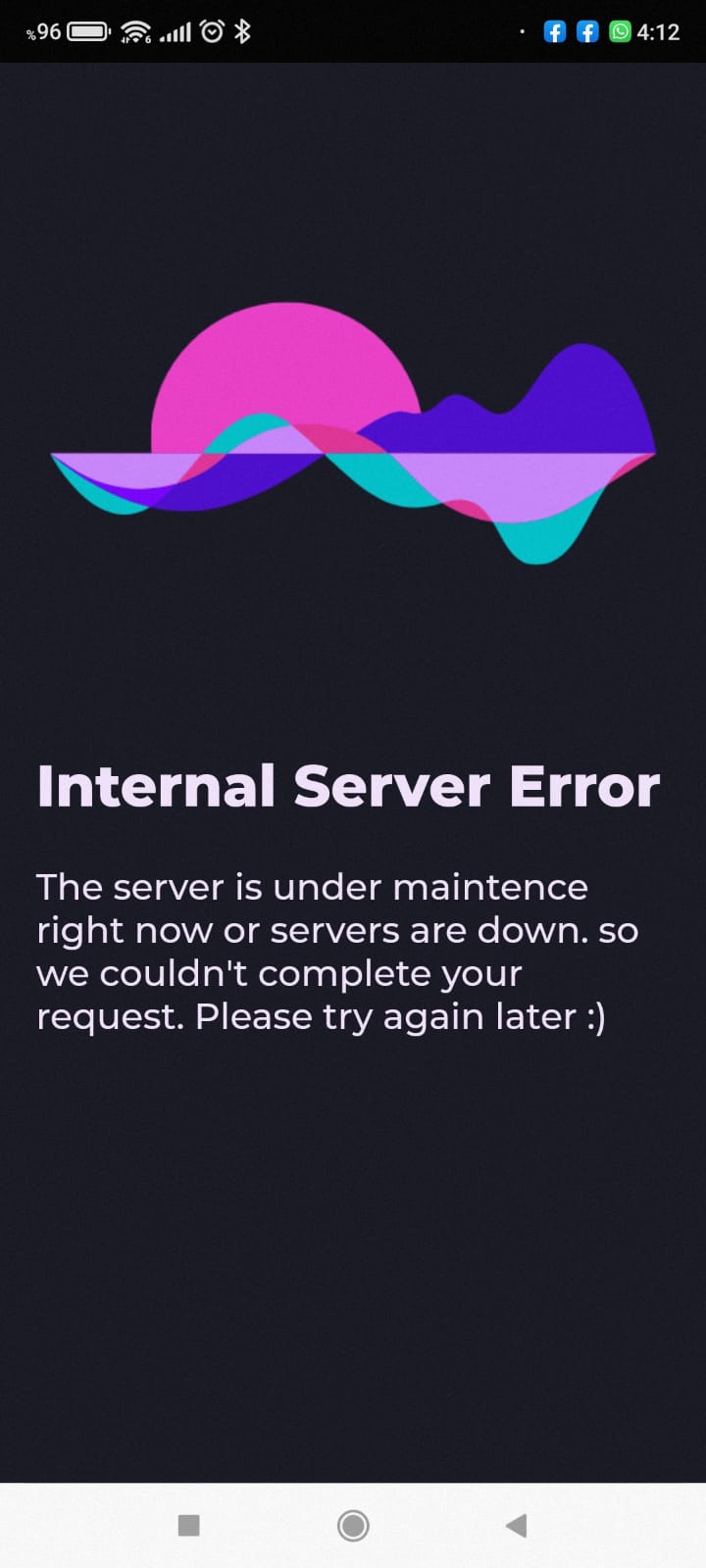
לאחר שמשתמש יעלה את התמונה שלו התמונה תחליף את תמונת הפנים שראינו קודם.

• 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.

import os

import subprocess

from time import sleep

from server import Server

def handle\_os\_error():

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

    result = result[71:].strip()

    os.system(f"tskill {result}")

def main():

    try:

        Server()

    except OSError:

        handle\_os\_error()

        sleep(2)

        Server()

if \_\_name\_\_ == "\_\_main\_\_":

    main()

### 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 \_\_init\_\_(self) -> None:

        with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server:

            server.bind(self.\_\_SERVER\_INFO)

            print(f"server ip - {self.\_\_SERVER\_INFO[0]}, server port - {self.\_\_SERVER\_INFO[1]}")

            print("jsut binded !")

            server.listen()

            print("i have an ear problem, but i can still listen :)")

            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()

            self.\_\_run()

    def \_\_del\_\_(self) -> None:

        self.\_\_should\_listen = False

    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

    def \_\_receive\_clients(self) -> None:

        while self.\_\_should\_listen:

            try:

                client, addr = self.\_\_server.accept()

                print("oh new client yee pee kay yay !")

                self.\_\_unhandled\_clients.append(client)

            except OSError:

                pass

    def \_\_create\_thread(self, client) -> None:

        thread = Thread(target=ClientHandler, args=(client, self.\_\_db\_manager, self.\_\_model\_lock, self.\_\_model))

        self.\_\_threads.append(thread)

        thread.start()

### 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.

    def \_\_init\_\_(self) -> None:

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

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

        self.\_\_setup\_db()

    def \_\_setup\_db(self):

        self.\_\_cursor.execute(

            """

        CREATE TABLE IF NOT EXISTS  users (

            "email" TEXT,

            "password"  TEXT,

            PRIMARY KEY("email")

        );

        """

        )

    def login(self, email: str, password: str) -> bool:

        check\_user = f"SELECT \* FROM users WHERE email = '{email}'  and password = '{password}'"

        self.\_\_cursor.execute(check\_user)

        ans = self.\_\_cursor.fetchall()

        return len(ans) == 1

    def signup(self, email: str, password: str) -> bool:

        check\_user = f"SELECT \* FROM users WHERE email = '{email}'"

        self.\_\_cursor.execute(check\_user)

        ans = self.\_\_cursor.fetchall()

        if len(ans) == 1:

            return False

        insert = f"INSERT INTO users VALUES ('{email}', '{password}');"

        self.\_\_cursor.execute(insert)

        self.\_\_conn.commit()

        return True

### 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 \_\_init\_\_(self, sock: socket.socket, db\_manager: tuple, model\_lock: Lock, model) -> None:

        self.\_\_sock = sock

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

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

        self.\_\_model = model

        self.\_\_handle\_client()

    def \_\_send\_message(self, message: str):

        try:

            full\_message = bytearray()

            full\_message.extend("1".encode())

            full\_message.extend((len(message)).to\_bytes(4, byteorder="little"))

            full\_message.extend(message.encode())

            self.\_\_sock.send(full\_message)

        except:

            pass

    def \_\_send\_ping(self):

        try:

            self.\_\_sock.send("2".encode())

            return True

        except:

            return False

    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

    @staticmethod

    def \_\_recvall(sock: socket.socket, length: int) -> str:

        data = bytearray()

        while len(data) < length:

            packet = sock.recv(1024)

            if not packet:

                pass

            data.extend(packet)

        return data

    def \_\_wait\_for\_lock(self, lock: Lock):

        while lock.locked():

            sleep(0.1)

    def \_\_login(self, data: bytearray) -> None:

        data = data.decode()

        email, password = data.split(",")

        is\_login = 0

        while type(is\_login) == int:

            self.\_\_wait\_for\_lock(self.\_\_db\_manager[1])

            try:

                self.\_\_db\_manager[1].acquire()

                is\_login = self.\_\_db\_manager[0].login(email, password)

            finally:

                self.\_\_db\_manager[1].release()

        self.\_\_send\_message(str(int(is\_login)))

    def \_\_signup(self, data: bytearray) -> None:

        data = data.decode()

        email, password = data.split(",")

        is\_registered = 0

        while type(is\_registered) == int:

            self.\_\_wait\_for\_lock(self.\_\_db\_manager[1])

            try:

                self.\_\_db\_manager[1].acquire()

                is\_registered = self.\_\_db\_manager[0].signup(email, password)

            finally:

                self.\_\_db\_manager[1].release()

        self.\_\_send\_message(str(int(is\_registered)))

    def \_\_find\_song(self, data: bytearray):

        emotion = 0

        while type(emotion) == int:

            self.\_\_wait\_for\_lock(self.\_\_model\_lock)

            try:

                self.\_\_model\_lock.acquire()

                emotion = self.\_\_find\_emotion(data)

            finally:

                self.\_\_model\_lock.release()

        song = self.\_\_match\_song\_to\_emotion(emotion)

        details = emotion + "," + ",".join(song)

        self.\_\_send\_message(details)

    def \_\_match\_song\_to\_emotion(self, emotion: str) -> list:

        songs = {

            "Angry": ["Metallica", "St. Anger", "https://www.youtube.com/watch?v=3rFoGVkZ29w&ab\_channel=Metallica"],

            "Disgust": [

                "ADELE",

                "Cold Shoulder",

                "https://www.youtube.com/watch?v=uGwH-x4VoH8&ab\_channel=XLRecordings",

            ],

            "Fear": ["Mariah Carey", "Hero", "https://www.youtube.com/watch?v=0IA3ZvCkRkQ&ab\_channel=MariahCareyVEVO"],

            "Happy": [

                "Justin Timberlake",

                "CAN'T STOP THE FEELING!",

                "https://www.youtube.com/watch?v=ru0K8uYEZWw&ab\_channel=justintimberlakeVEVO",

            ],

            "Sad": [

                "We The Kings",

                "Sad Song",

                "https://www.youtube.com/watch?v=BZsXcc\_tC-o&ab\_channel=WeTheKingsVEVO",

            ],

            "Surprise": [

                "The Rolling Stones",

                "Surprise, Surprise",

                "https://www.youtube.com/watch?v=whfzPSgUglk&ab\_channel=ABKCOVEVO",

            ],

            "Neutral": [

                "The Weeknd",

                "Blinding Lights",

                "https://www.youtube.com/watch?v=fHI8X4OXluQ&ab\_channel=TheWeekndVEVO",

            ],

        }

        return songs[emotion]

    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 ספלאש

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:song\_feeler/resources/helper.dart';

import 'package:song\_feeler/resources/socket.dart';

// ignore: must\_be\_immutable

class Splash extends StatefulWidget {

  final double widthRatio;

  final double heightRatio;

  bool connected = false;

  Splash({Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  State<Splash> createState() => \_SplashState();

}

class \_SplashState extends State<Splash> with WidgetsBindingObserver {

  var socket = MSocket();

  @override

  void initState() {

    super.initState();

    WidgetsBinding.instance?.addObserver(this);

    Future.delayed(Duration.zero, () async {

      do {

        widget.connected = await socket.connectServer(context);

      } while (!widget.connected);

      var path = await configFilePath;

      if (await File(path).exists()) {

        final file = await configFile;

        final content = await file.readAsString();

        String reply = await socket.sendAndWait(context, content, 1);

        if (reply == "1") {

          MSocket.pushReplacement(context, '/menu', socket);

        } else {

          MSocket.pushReplacement(context, '/login', socket);

        }

      } else {

        MSocket.pushReplacement(context, '/login', socket);

      }

    });

  }

  @override

  void dispose() {

    WidgetsBinding.instance?.removeObserver(this);

    super.dispose();

  }

  @override

  void didChangeAppLifecycleState(AppLifecycleState state) {

    super.didChangeAppLifecycleState(state);

    if (state == AppLifecycleState.paused) {

      socket.close();

      SystemNavigator.pop();

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

        body: Column(

          crossAxisAlignment: CrossAxisAlignment.center,

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            Padding(

              padding: EdgeInsets.symmetric(horizontal: 12 \* widget.widthRatio),

              child: Image.asset('assets/SongFeeler.png'),

            ),

            SizedBox(

              height: 120 \* widget.heightRatio,

            ),

            Padding(

                padding:

                    EdgeInsets.symmetric(horizontal: 60 \* widget.widthRatio),

                child: Image.asset('assets/loading.gif')),

          ],

        ));

  }

}

### מסך שגיאת התחברות

import 'package:flutter/material.dart';

class ConnectionE extends StatelessWidget {

  final double widthRatio;

  final double heightRatio;

  const ConnectionE({Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  Widget build(BuildContext context) {

    return SafeArea(

        child: Scaffold(

      backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

      body: Padding(

        padding: const EdgeInsets.symmetric(horizontal: 20),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          mainAxisAlignment: MainAxisAlignment.start,

          children: <Widget>[

            SizedBox(

              height: 30 \* heightRatio,

            ),

            Align(

              alignment: Alignment.center,

              child: Padding(

                padding: EdgeInsets.zero,

                child: Image.asset('assets/Logo.png'),

              ),

            ),

            Text(

              "Internal Server Error",

              style: TextStyle(

                  fontSize: 32 \* heightRatio,

                  fontFamily: 'Montserrat',

                  fontWeight: FontWeight.w800,

                  color: const Color.fromRGBO(244, 231, 255, 1)),

            ),

            SizedBox(

              height: 25 \* heightRatio,

            ),

            Text(

              "The server is under maintence right now or servers are down. so we couldn't complete your request. Please try again later :)",

              style: TextStyle(

                  fontSize: 20 \* heightRatio,

                  fontFamily: 'Montserrat',

                  fontWeight: FontWeight.w500,

                  color: const Color.fromRGBO(244, 231, 255, 1)),

            ),

            SizedBox(

              height: 80 \* heightRatio,

            ),

          ],

        ),

      ),

      floatingActionButton: Padding(

        padding: EdgeInsets.only(bottom: 15 \* heightRatio),

        child: TextButton.icon(

            onPressed: () {

              Navigator.pop(context

              );

            },

            style: TextButton.styleFrom(

                shape: const CircleBorder(),

                backgroundColor: const Color.fromRGBO(244, 231, 255, 1),

                padding: EdgeInsets.fromLTRB(6 \* widthRatio, 0, 0, 0),

                fixedSize: Size(65 \* heightRatio, 65 \* widthRatio),

                splashFactory: NoSplash.splashFactory),

            icon: Icon(

              Icons.refresh\_rounded,

              size: 48 \* heightRatio,

              color: const Color.fromRGBO(0, 196, 204, 1),

            ),

            label: const Text("")),

      ),

    ));

  }

}

### מסך התחברות

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:song\_feeler/resources/helper.dart';

import 'package:song\_feeler/resources/socket.dart';

import 'package:song\_feeler/resources/login\_signup.dart';

import 'package:font\_awesome\_flutter/font\_awesome\_flutter.dart';

// ignore: must\_be\_immutable

class Login extends StatefulWidget {

  final double widthRatio;

  final double heightRatio;

  bool ignore = false;

  bool isPassValid = true;

  bool isEmailValid = true;

  int i = 0;

  Login({Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  State<Login> createState() => \_LoginState();

}

class \_LoginState extends State<Login> with WidgetsBindingObserver {

  late MSocket socket;

  Completer completer = Completer();

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  @override

  void initState() {

    super.initState();

    WidgetsBinding.instance?.addObserver(this);

  }

  @override

  void dispose() {

    WidgetsBinding.instance?.removeObserver(this);

    super.dispose();

  }

  @override

  void didChangeAppLifecycleState(AppLifecycleState state) {

    super.didChangeAppLifecycleState(state);

    if (completer.isCompleted) {

      if (state == AppLifecycleState.paused) {

        socket.close();

        SystemNavigator.pop();

      }

    }

  }

  void onPressed(MSocket socket) async {

    widget.ignore = false;

    if (!widget.ignore) {

      widget.i += 1;

      setState(() {});

      widget.isEmailValid = getEmailError() == "";

      widget.isPassValid = getPasswordError() == "";

      if (widget.isEmailValid && widget.isPassValid) {

        try {

          await loginRequest(socket);

        } catch (error) {

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

        }

      }

    }

  }

  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);

  }

  Future<void> loginRequest(MSocket socket) async {

    String content = '${emailController.text},${passwordController.text}';

    String reply = await socket.sendAndWait(context, content, 1);

    if (reply == "1") {

      writeConfig(content);

      MSocket.pushReplacement(context, '/menu', socket);

    } else if (reply == "2") {

    } else {

      MSocket.showError(context, "Login Error",

          "email or password are incorrect, please try again :)");

    }

  }

  String getEmailError() {

    if (emailController.text.isEmpty) {

      return "Email field is required";

    } else if (!\_isEmailValid(emailController.text)) {

      return "Email is not valid";

    }

    return "";

  }

  String getPasswordError() {

    if (passwordController.text.isEmpty) {

      return "Password field is required";

    } else if (\_isPasswordValid(passwordController.text)) {

      return "Password is not valid";

    } else if (passwordController.text.length < 8) {

      return "Password length must be at least 8 characters";

    }

    return "";

  }

  @override

  Widget build(BuildContext context) {

    final arguments = (ModalRoute.of(context)?.settings.arguments ??

        <String, dynamic>{}) as Map;

    socket = arguments['Socket'];

    if(!completer.isCompleted) {

      completer.complete();

    }

    Email email = Email(

      label: 'Email',

      ratio: widget.heightRatio,

      list: [FilteringTextInputFormatter.deny(RegExp('[ ]'))],

      controller: emailController,

      isValid: widget.isEmailValid,

    );

    Password password = Password(

      ratio: widget.heightRatio,

      controller: passwordController,

      isValid: widget.isPassValid,

    );

    return Scaffold(

      resizeToAvoidBottomInset: false,

      backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

      body: SingleChildScrollView(

          reverse: true,

          child: SafeArea(

            child: Container(

              margin: EdgeInsets.symmetric(

                horizontal: 24 \* widget.widthRatio,

              ),

              child: Column(

                crossAxisAlignment: CrossAxisAlignment.start,

                mainAxisAlignment: MainAxisAlignment.center,

                children: [

                  SizedBox(

                    height: 50 \* widget.heightRatio,

                  ),

                  BigText(

                    text: 'Welcome',

                    ratio: widget.heightRatio,

                  ),

                  BigText(

                    text: 'Back',

                    ratio: widget.heightRatio,

                  ),

                  SizedBox(

                    height: 30 \* widget.heightRatio,

                  ),

                  SmallText(

                    text: 'Login with one of the following options.',

                    ratio: widget.heightRatio,

                  ),

                  SizedBox(

                    height: 40 \* widget.heightRatio,

                  ),

                  email,

                  showFieldError(getEmailError(), widget.isEmailValid),

                  password,

                  showFieldError(getPasswordError(), widget.isPassValid),

                  Row(

                    children: [

                      Expanded(

                          child: StyledButton(

                        ratio: widget.heightRatio,

                        text: 'Login',

                        onPressed: () => onPressed(socket),

                      )),

                    ],

                  ),

                  Divider(

                      height: 80 \* widget.heightRatio,

                      thickness: 1,

                      color: const Color.fromRGBO(98, 114, 130, 1)),

                  Row(

                    children: [

                      StyledIconButton(

                          icon: Icon(FontAwesomeIcons.google,

                              size: 32 \* widget.heightRatio,

                              color: const Color.fromRGBO(238, 243, 245, 1)),

                          ratio: widget.heightRatio,

                          text: const Text('')),

                      SizedBox(width: 20 \* widget.widthRatio),

                      StyledIconButton(

                        icon: Icon(Icons.apple,

                            size: 42 \* widget.heightRatio,

                            color: const Color.fromRGBO(238, 243, 245, 1)),

                        ratio: widget.heightRatio,

                        text: const Text(''),

                      ),

                    ],

                  ),

                  SizedBox(

                    height: 20 \* widget.heightRatio,

                  ),

                  Row(

                    mainAxisAlignment: MainAxisAlignment.center,

                    children: [

                      SmallText(

                        text: 'Dont have an account?',

                        ratio: widget.heightRatio,

                      ),

                      SmallTextButton(

                        text: 'Sign up',

                        ratio: widget.heightRatio,

                        onPreesed: () =>

                            MSocket.pushReplacement(context, "/signup", socket),

                      )

                    ],

                  ),

                  Padding(

                      padding: EdgeInsets.only(

                          bottom: MediaQuery.of(context).viewInsets.bottom))

                ],

              ),

            ),

          )),

    );

  }

  Padding showFieldError(String text, bool isValid) {

    var left = 5 \* widget.widthRatio;

    var top = 6 \* widget.heightRatio;

    var bottom = 10 \* widget.heightRatio;

    if (widget.i == 0) {

      text = "";

    }

    if (text == "") {

      left = 0;

      top = 0;

      bottom = 0;

    }

    return Padding(

        padding: EdgeInsets.fromLTRB(left, top, 0, bottom),

        child: Text(

          text,

          style: const TextStyle(color: Colors.red),

        ));

  }

}

### מסך הירשמות

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:font\_awesome\_flutter/font\_awesome\_flutter.dart';

import 'package:song\_feeler/resources/login\_signup.dart';

import 'package:song\_feeler/resources/helper.dart';

import 'package:song\_feeler/resources/socket.dart';

import 'package:flutter/services.dart';

// ignore: must\_be\_immutable

class Signup extends StatefulWidget {

  final double widthRatio;

  final double heightRatio;

  bool ignore = false;

  bool isPassValid = true;

  bool isEmailValid = true;

  int i = 0;

  Signup({Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  State<Signup> createState() => \_SignupState();

}

class \_SignupState extends State<Signup> with WidgetsBindingObserver {

  late MSocket socket;

  Completer completer = Completer();

  final emailController = TextEditingController();

  final passwordController = TextEditingController();

  @override

  void initState() {

    super.initState();

    WidgetsBinding.instance?.addObserver(this);

  }

  @override

  void dispose() {

    WidgetsBinding.instance?.removeObserver(this);

    super.dispose();

  }

  @override

  void didChangeAppLifecycleState(AppLifecycleState state) {

    super.didChangeAppLifecycleState(state);

    if (completer.isCompleted) {

      if (state == AppLifecycleState.paused) {

        socket.close();

        SystemNavigator.pop();

      }

    }

  }

  void onPressed(MSocket socket) async {

    if (!widget.ignore) {

      widget.i += 1;

      setState(() {});

      widget.isEmailValid = getEmailError() == "";

      widget.isPassValid = getPasswordError() == "";

      widget.ignore = true;

      if (widget.isEmailValid && widget.isPassValid) {

        try {

          await signupRequest(socket);

        } catch (error) {

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

        }

      }

      widget.ignore = false;

    }

  }

  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);

  }

  String getEmailError() {

    if (emailController.text.isEmpty) {

      return "Email field is required";

    } else if (!\_isEmailValid(emailController.text)) {

      return "Email is not valid";

    }

    return "";

  }

  String getPasswordError() {

    if (passwordController.text.isEmpty) {

      return "Password field is required";

    } else if (\_isPasswordValid(passwordController.text)) {

      return "Password is not valid";

    } else if (passwordController.text.length < 8) {

      return "Password length must be at least 8 characters";

    }

    return "";

  }

  Future<void> signupRequest(MSocket socket) async {

    String content = '${emailController.text},${passwordController.text}';

    String reply = await socket.sendAndWait(context, content, 2);

    if (reply == "1") {

      writeConfig(content);

      Navigator.pushReplacementNamed(context, '/menu',

          arguments: {"Socket": socket});

    } else {

      MSocket.showError(context, "Signup Error",

          "email is already exists, please try again :)");

    }

  }

  @override

  Widget build(BuildContext context) {

    final arguments = (ModalRoute.of(context)?.settings.arguments ??

        <String, dynamic>{}) as Map;

    socket = arguments['Socket'];

    if(!completer.isCompleted) {

      completer.complete();

    }

    Email email = Email(

      label: 'Email',

      ratio: widget.heightRatio,

      list: [FilteringTextInputFormatter.deny(RegExp('[ ]'))],

      controller: emailController,

      isValid: widget.isEmailValid,

    );

    Password password = Password(

        ratio: widget.heightRatio,

        controller: passwordController,

        isValid: widget.isPassValid);

    return Scaffold(

      backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

      resizeToAvoidBottomInset: false,

      body: SingleChildScrollView(

        reverse: true,

        child: SafeArea(

            child: Container(

          margin: EdgeInsets.symmetric(horizontal: 24 \* widget.widthRatio),

          child: Column(

            crossAxisAlignment: CrossAxisAlignment.start,

            children: [

              SizedBox(

                height: 50 \* widget.heightRatio,

              ),

              BigText(

                text: 'Create',

                ratio: widget.heightRatio,

              ),

              BigText(

                text: 'Account',

                ratio: widget.heightRatio,

              ),

              SizedBox(

                height: 30 \* widget.heightRatio,

              ),

              SmallText(

                  text: 'Signup with one of the following options.',

                  ratio: widget.heightRatio),

              SizedBox(

                height: 40 \* widget.heightRatio,

              ),

              email,

              showFieldError(getEmailError(), widget.isEmailValid),

              password,

              showFieldError(getPasswordError(), widget.isPassValid),

              Row(

                children: [

                  Expanded(

                    child: StyledButton(

                      text: 'Sign up',

                      ratio: widget.heightRatio,

                      onPressed: () => onPressed(socket),

                    ),

                  ),

                ],

              ),

              Divider(

                  height: 80 \* widget.heightRatio,

                  thickness: 1,

                  color: const Color.fromRGBO(98, 114, 130, 1)),

              Row(

                children: [

                  StyledIconButton(

                    icon: Icon(FontAwesomeIcons.google,

                        size: 32 \* widget.heightRatio,

                        color: const Color.fromRGBO(238, 243, 245, 1)),

                    ratio: widget.heightRatio,

                    text: const Text(''),

                  ),

                  SizedBox(width: 20 \* widget.widthRatio),

                  StyledIconButton(

                    icon: Icon(Icons.apple,

                        size: 42 \* widget.heightRatio,

                        color: const Color.fromRGBO(238, 243, 245, 1)),

                    ratio: widget.heightRatio,

                    text: const Text(''),

                  ),

                ],

              ),

              SizedBox(

                height: 20 \* widget.heightRatio,

              ),

              Row(

                mainAxisAlignment: MainAxisAlignment.center,

                children: [

                  SmallText(

                    text: 'Already have an account?',

                    ratio: widget.heightRatio,

                  ),

                  SmallTextButton(

                      text: 'Login',

                      ratio: widget.heightRatio,

                      onPreesed: () =>

                          MSocket.pushReplacement(context, "/login", socket))

                ],

              ),

              Padding(

                  padding: EdgeInsets.only(

                      bottom: MediaQuery.of(context).viewInsets.bottom))

            ],

          ),

        )),

      ),

    );

  }

  Padding showFieldError(String text, bool isValid) {

    var left = 5 \* widget.widthRatio;

    var top = 6 \* widget.heightRatio;

    var bottom = 10 \* widget.heightRatio;

    if (widget.i == 0) {

      text = "";

    }

    if (text == "") {

      left = 0;

      top = 0;

      bottom = 0;

    }

    return Padding(

        padding: EdgeInsets.fromLTRB(left, top, 0, bottom),

        child: Text(

          text,

          style: const TextStyle(color: Colors.red),

        ));

  }

}

### מסך בית

import 'dart:io';

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:song\_feeler/resources/menu.dart';

import 'package:song\_feeler/resources/socket.dart';

// ignore: must\_be\_immutable

class Menu extends StatefulWidget {

  final double widthRatio;

  final double heightRatio;

  bool ignore = false;

  Menu({

    Key? key,

    required this.widthRatio,

    required this.heightRatio,

  }) : super(key: key);

  @override

  State<Menu> createState() => \_MenuState();

}

class \_MenuState extends State<Menu> with WidgetsBindingObserver {

  Completer completer = Completer();

  late MSocket socket;

  File? image;

  @override

  void initState() {

    super.initState();

    WidgetsBinding.instance?.addObserver(this);

  }

  @override

  void dispose() {

    WidgetsBinding.instance?.removeObserver(this);

    super.dispose();

  }

  @override

  void didChangeAppLifecycleState(AppLifecycleState state) {

    super.didChangeAppLifecycleState(state);

    if (completer.isCompleted) {

      if (state == AppLifecycleState.detached) {

        socket.close();

        SystemNavigator.pop();

      }

    }

  }

  Future<void> pickimage(MSocket socket) async {

    try {

      final image = await ImagePicker().pickImage(source: ImageSource.gallery);

      if (image == null) return;

      final tempImage = File(image.path);

      setState((() => this.image = tempImage));

      // ignore: empty\_catches

    } on PlatformException {}

  }

  void onFeelMusic(MSocket socket) async {

    if (!widget.ignore) {

      widget.ignore = true;

      if (image == null) {

        MSocket.showError(context, "Image Error",

            "image can't be empty, u need to choose one :)");

      } else {

        MSocket.pushReplacementWithArgs(

            context, "/loading", {"Socket": socket, "Image": image});

      }

      widget.ignore = false;

    }

  }

  @override

  Widget build(BuildContext context) {

    final arguments = (ModalRoute.of(context)?.settings.arguments ??

        <String, dynamic>{}) as Map;

    socket = arguments['Socket'];

    if (!completer.isCompleted) {

      completer.complete();

    }

    return Scaffold(

      backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

      body: SingleChildScrollView(

        reverse: true,

        child: SafeArea(

          child: Container(

            margin: EdgeInsets.symmetric(

              horizontal: 24 \* widget.widthRatio,

            ),

            child: Column(

              crossAxisAlignment: CrossAxisAlignment.start,

              children: [

                SizedBox(

                  height: 60 \* widget.heightRatio,

                ),

                BigText(ratio: widget.heightRatio),

                SizedBox(

                  height: 20 \* widget.heightRatio,

                ),

                SmallText(ratio: widget.heightRatio),

                SizedBox(

                  height: 40 \* widget.heightRatio,

                ),

                image == null

                    ? StlyedIcon(

                        heightRatio: widget.heightRatio,

                        widthRatio: widget.widthRatio,

                        onPressed: pickimage,

                        socket: socket,

                      )

                    : Center(

                        child: StyledImage(

                          heightRatio: widget.heightRatio,

                          widthRatio: widget.widthRatio,

                          image: image,

                        ),

                      ),

                SizedBox(

                  height: 30 \* widget.heightRatio,

                ),

                StyledTextButton(

                  heightRatio: widget.heightRatio,

                  widthRatio: widget.widthRatio,

                  onPressed: onFeelMusic,

                  socket: socket,

                ),

              ],

            ),

          ),

        ),

      ),

      bottomNavigationBar: StyledAppBar(

        ratio: widget.heightRatio,

        page: "/menu",

        socket: socket,

      ),

    );

  }

}

### מסך טעינה

import 'dart:typed\_data';

import 'package:flutter/services.dart';

import 'package:flutter/material.dart';

import 'package:song\_feeler/resources/socket.dart';

class Loading extends StatefulWidget {

  final double widthRatio;

  final double heightRatio;

  const Loading({Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  State<Loading> createState() => \_LoadingState();

}

class \_LoadingState extends State<Loading> with WidgetsBindingObserver {

  Map<dynamic, dynamic>? arguments;

  @override

  void initState() {

    super.initState();

    Future.delayed(Duration.zero, () async {

      arguments = (ModalRoute.of(context)?.settings.arguments ??

          <String, dynamic>{}) as Map;

      WidgetsBinding.instance?.addObserver(this);

      final socket = arguments!['Socket'];

      final image = arguments!['Image'];

      final Uint8List data = await image!.readAsBytes();

      final reply = await socket.sendAndWaitForImage(context, data, 3);

      if (reply != "2") {

        MSocket.pushReplacementWithArgs(context, "/final",

            {"Socket": socket, "Image": image, "Reply": reply});

      }

    });

  }

  @override

  void dispose() {

    WidgetsBinding.instance?.removeObserver(this);

    super.dispose();

  }

  @override

  void didChangeAppLifecycleState(AppLifecycleState state) {

    super.didChangeAppLifecycleState(state);

    if (state == AppLifecycleState.paused) {

      final socket = arguments!['Socket'];

      socket.close();

      SystemNavigator.pop();

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

        backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

        body: Column(

          crossAxisAlignment: CrossAxisAlignment.center,

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            Padding(

              padding: EdgeInsets.symmetric(horizontal: 12 \* widget.widthRatio),

              child: Image.asset('assets/Symbol.png'),

            ),

            SizedBox(

              height: 10 \* widget.heightRatio,

            ),

            Text("Processing Image...",

                style: TextStyle(

                    fontSize: 36 \* widget.heightRatio,

                    fontFamily: 'Montserrat',

                    fontWeight: FontWeight.w800,

                    color: const Color.fromRGBO(0, 196, 204, 1))),

            SizedBox(

              height: 5 \* widget.heightRatio,

            ),

            Text("Just Few More Seconds",

                style: TextStyle(

                    fontSize: 26 \* widget.heightRatio,

                    fontFamily: 'Montserrat',

                    fontWeight: FontWeight.w800,

                    color: const Color.fromRGBO(228, 51, 151, 1))),

            SizedBox(

              height: 140 \* widget.heightRatio,

            ),

            Padding(

                padding:

                    EdgeInsets.symmetric(horizontal: 60 \* widget.widthRatio),

                child: Image.asset('assets/loading.gif')),

          ],

        ));

  }

}

### מסך שגיאת שרת

import 'package:flutter/material.dart';

class ServerDownE extends StatelessWidget {

  final double widthRatio;

  final double heightRatio;

  const ServerDownE(

      {Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  Widget build(BuildContext context) {

    final arguments = (ModalRoute.of(context)?.settings.arguments ??

        <String, dynamic>{}) as Map;

    final socket = arguments['Socket'];

    socket.close();

    return SafeArea(

        child: Scaffold(

      backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

      body: Padding(

        padding: const EdgeInsets.symmetric(horizontal: 20),

        child: Column(

          crossAxisAlignment: CrossAxisAlignment.start,

          mainAxisAlignment: MainAxisAlignment.start,

          children: <Widget>[

            SizedBox(

              height: 30 \* heightRatio,

            ),

            Align(

              alignment: Alignment.center,

              child: Padding(

                padding: EdgeInsets.zero,

                child: Image.asset('assets/Logo.png'),

              ),

            ),

            Text(

              "Internal Server Error",

              style: TextStyle(

                  fontSize: 32 \* heightRatio,

                  fontFamily: 'Montserrat',

                  fontWeight: FontWeight.w800,

                  color: const Color.fromRGBO(244, 231, 255, 1)),

            ),

            SizedBox(

              height: 25 \* heightRatio,

            ),

            Text(

              "The server is under maintence right now or servers are down. so we couldn't complete your request. Please try again later :)",

              style: TextStyle(

                  fontSize: 20 \* heightRatio,

                  fontFamily: 'Montserrat',

                  fontWeight: FontWeight.w500,

                  color: const Color.fromRGBO(244, 231, 255, 1)),

            ),

            SizedBox(

              height: 80 \* heightRatio,

            ),

          ],

        ),

      ),

    ));

  }

}

### מסך תוצאה

import 'dart:async';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:font\_awesome\_flutter/font\_awesome\_flutter.dart';

import 'package:song\_feeler/resources/menu.dart';

import 'package:song\_feeler/resources/socket.dart';

import 'package:url\_launcher/url\_launcher\_string.dart';

class Final extends StatefulWidget {

  final double widthRatio;

  final double heightRatio;

  const Final({Key? key, required this.widthRatio, required this.heightRatio})

      : super(key: key);

  @override

  State<Final> createState() => \_FinalState();

}

class \_FinalState extends State<Final> with WidgetsBindingObserver {

  late MSocket socket;

  Completer completer = Completer();

  @override

  void initState() {

    super.initState();

    WidgetsBinding.instance?.addObserver(this);

  }

  @override

  void dispose() {

    WidgetsBinding.instance?.removeObserver(this);

    super.dispose();

  }

  @override

  void didChangeAppLifecycleState(AppLifecycleState state) {

    super.didChangeAppLifecycleState(state);

    if (completer.isCompleted) {

      if (state == AppLifecycleState.paused) {

        socket.close();

        SystemNavigator.pop();

      }

    }

  }

  @override

  Widget build(BuildContext context) {

    final arguments = (ModalRoute.of(context)?.settings.arguments ??

        <String, dynamic>{}) as Map;

    socket = arguments['Socket'];

    if (!completer.isCompleted) {

      completer.complete();

    }

    final image = arguments['Image'];

    final reply = arguments['Reply'];

    var list = reply.split(",");

    return Scaffold(

      backgroundColor: const Color.fromRGBO(22, 22, 34, 1),

      body: Column(

        mainAxisAlignment: MainAxisAlignment.end,

        children: [

          Align(

            alignment: Alignment.center,

            child: SizedBox(

              height: 520 \* widget.heightRatio,

              width: 350 \* widget.widthRatio,

              child: Stack(

                alignment: Alignment.center,

                children: [

                  Container(

                    height: 520 \* widget.heightRatio,

                    width: 350 \* widget.widthRatio,

                    decoration: BoxDecoration(

                        image: DecorationImage(

                          fit: BoxFit.fill,

                          image: FileImage(image!),

                        ),

                        color: const Color.fromRGBO(59, 60, 80, 1),

                        borderRadius: BorderRadius.all(

                            Radius.circular(20.0 \* widget.heightRatio)),

                        boxShadow: [

                          BoxShadow(

                            color: const Color.fromRGBO(244, 231, 255, 1)

                                .withOpacity(0.1),

                            spreadRadius: 10 \* widget.heightRatio,

                            blurRadius: 20 \* widget.heightRatio,

                          ),

                        ]),

                  ),

                  Align(

                    alignment: Alignment.bottomLeft,

                    child: Padding(

                      padding: EdgeInsets.only(

                          left: 15 \* widget.widthRatio,

                          bottom: 15 \* widget.heightRatio),

                      child: Column(

                        mainAxisAlignment: MainAxisAlignment.end,

                        crossAxisAlignment: CrossAxisAlignment.start,

                        children: [

                          Text(

                            "Emotion: ${list[0]}",

                            style: TextStyle(

                                fontSize: 17 \* widget.heightRatio,

                                fontFamily: 'Montserrat',

                                fontWeight: FontWeight.w700,

                                color: const Color.fromRGBO(238, 243, 245, 1)),

                          ),

                          Text(

                            "Singer: ${list[1]}",

                            style: TextStyle(

                                fontSize: 17 \* widget.heightRatio,

                                fontFamily: 'Montserrat',

                                fontWeight: FontWeight.w700,

                                color: const Color.fromRGBO(238, 243, 245, 1)),

                          ),

                          Text(

                            "Song: ${list[2]}",

                            style: TextStyle(

                                fontSize: 17 \* widget.heightRatio,

                                fontFamily: 'Montserrat',

                                fontWeight: FontWeight.w700,

                                color: const Color.fromRGBO(238, 243, 245, 1)),

                          ),

                        ],

                      ),

                    ),

                  ),

                ],

              ),

            ),

          ),

          SizedBox(

            height: 20 \* widget.heightRatio,

          ),

          Row(

            mainAxisAlignment: MainAxisAlignment.center,

            children: [

              Align(

                alignment: Alignment.bottomRight,

                child: TextButton.icon(

                    onPressed: () async {

                      String url = list[3];

                      try {

                        await launchUrlString(url,

                            mode: LaunchMode.externalApplication);

                      }

                      // ignore: empty\_catches

                      catch (error) {

                        MSocket.showError(context, "Song Error",

                            "we are sorry but something happend with the link :(");

                      }

                    },

                    style: TextButton.styleFrom(

                      padding: EdgeInsets.symmetric(

                          horizontal: 18 \* widget.widthRatio,

                          vertical: 15 \* widget.heightRatio),

                      backgroundColor: const Color.fromRGBO(40, 39, 57, 1),

                      splashFactory: NoSplash.splashFactory,

                      shape: RoundedRectangleBorder(

                        borderRadius: BorderRadius.all(

                          Radius.circular(20 \* widget.heightRatio),

                        ),

                      ),

                    ),

                    icon: Icon(

                      FontAwesomeIcons.youtube,

                      color: Colors.red,

                      size: 40 \* widget.heightRatio,

                    ),

                    label: Text("\t\tGo & Listen On Youtube",

                        style: TextStyle(

                            fontSize: 22 \* widget.heightRatio,

                            fontFamily: 'Montserrat',

                            fontWeight: FontWeight.w700,

                            color: const Color.fromRGBO(206, 207, 222, 1)))),

              ),

            ],

          ),

          SizedBox(

            height: 70 \* widget.heightRatio,

          )

        ],

      ),

      bottomNavigationBar: StyledAppBar(

        ratio: widget.heightRatio,

        page: "/menu",

        socket: socket,

      ),

    );

  }

}

### Helper.dart

קובץ הנועד לעזור לקרוא ולכתוב לקבצים

import 'dart:io';

import 'dart:convert';

import 'package:path\_provider/path\_provider.dart';

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.

import 'package:flutter/material.dart';

import 'package:font\_awesome\_flutter/font\_awesome\_flutter.dart';

import 'package:flutter/services.dart';

// ignore: non\_constant\_identifier\_names

const Color fieldColor = Color.fromRGBO(40, 39, 57, 1);

const Color buttonColor = Color.fromRGBO(59, 60, 80, 1);

class SmallText extends StatelessWidget {

  final String text;

  final double ratio;

  const SmallText({Key? key, required this.text, required this.ratio})

      : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Text(text,

        style: TextStyle(

            fontSize: 16 \* ratio,

            fontFamily: 'Montserrat',

            fontWeight: FontWeight.w500,

            color: const Color.fromRGBO(188, 195, 207, 1)));

  }

}

class BigText extends StatelessWidget {

  final String text;

  final double ratio;

  const BigText({

    Key? key,

    required this.text,

    required this.ratio,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Text(

      text,

      style: TextStyle(

          fontSize: 54 \* ratio,

          fontFamily: 'Montserrat',

          fontWeight: FontWeight.w900,

          letterSpacing: 1.1,

          color: const Color.fromRGBO(238, 243, 245, 1)),

    );

  }

}

// ignore: must\_be\_immutable

class Email extends StatefulWidget {

  final String label;

  final double ratio;

  final List<TextInputFormatter> list;

  final TextEditingController controller;

  final bool isValid;

  const Email({

    Key? key,

    required this.label,

    required this.ratio,

    required this.list,

    required this.controller,

    required this.isValid,

  }) : super(key: key);

  @override

  State<Email> createState() => \_EmailState();

}

class \_EmailState extends State<Email> {

  @override

  Widget build(BuildContext context) {

    Color color =

        widget.isValid ? const Color.fromRGBO(238, 243, 245, 1) : Colors.red;

    return SizedBox(

      height: 60 \* widget.ratio,

      child: TextFormField(

        controller: widget.controller,

        style: const TextStyle(color: Color.fromRGBO(238, 243, 245, 1)),

        textInputAction: TextInputAction.next,

        inputFormatters: widget.list,

        decoration: InputDecoration(

          fillColor: fieldColor,

          filled: true,

          enabledBorder: \_BuildBorder(color, 0.2),

          focusedBorder: \_BuildBorder(color, 0.6),

          disabledBorder: \_BuildBorder(color, 0.6),

          label: Text("\t\t${widget.label}",

              style: const TextStyle(color: Color.fromRGBO(206, 207, 222, 1))),

        ),

      ),

    );

  }

  // ignore: non\_constant\_identifier\_names

  OutlineInputBorder \_BuildBorder(Color color, double width) {

    return OutlineInputBorder(

        borderRadius: BorderRadius.circular(20 \* widget.ratio),

        borderSide: BorderSide(color: color, width: width));

  }

}

// ignore: must\_be\_immutable

class Password extends StatefulWidget {

  final double ratio;

  final TextEditingController controller;

  final bool isValid;

  const Password({

    Key? key,

    required this.ratio,

    required this.controller,

    required this.isValid,

  }) : super(key: key);

  @override

  State<Password> createState() => \_PasswordState();

}

class \_PasswordState extends State<Password> {

  bool click = true;

  IconData isClicked() {

    return (click == false) ? FontAwesomeIcons.eye : FontAwesomeIcons.eyeSlash;

  }

  @override

  Widget build(BuildContext context) {

    Color color =

        widget.isValid ? const Color.fromRGBO(238, 243, 245, 1) : Colors.red;

    return SizedBox(

      height: 60 \* widget.ratio,

      child: TextFormField(

        obscureText: click,

        controller: widget.controller,

        style: const TextStyle(color: Color.fromRGBO(238, 243, 245, 1)),

        textInputAction: TextInputAction.done,

        inputFormatters: [

          LengthLimitingTextInputFormatter(16),

          FilteringTextInputFormatter.deny(RegExp('[ ]')),

        ],

        decoration: InputDecoration(

          suffix: Container(

            padding: EdgeInsets.zero,

            child: TextButton.icon(

              onPressed: () {

                setState(() {

                  click = !click;

                });

              },

              icon: Icon(

                isClicked(),

                color: const Color.fromRGBO(98, 114, 130, 1),

                size: 21 \* widget.ratio,

              ),

              label: const Text(''),

              style:

                  TextButton.styleFrom(splashFactory: NoSplash.splashFactory),

            ),

          ),

          fillColor: fieldColor,

          filled: true,

          enabledBorder: OutlineInputBorder(

              borderRadius: BorderRadius.circular(20 \* widget.ratio),

              borderSide: BorderSide(

                  color: color, width: 0.2)),

          focusedBorder: OutlineInputBorder(

              borderRadius: BorderRadius.circular(20 \* widget.ratio),

              borderSide: BorderSide(

                  color: color, width: 0.6)),

          disabledBorder: OutlineInputBorder(

              borderRadius: BorderRadius.circular(20 \* widget.ratio),

              borderSide: BorderSide(

                  color: color, width: 0.6)),

          label: const Text('\t\tPassword',

              style: TextStyle(color: Color.fromRGBO(206, 207, 222, 1))),

        ),

      ),

    );

  }

}

class SmallTextButton extends StatelessWidget {

  final String text;

  final double ratio;

  final VoidCallback onPreesed;

  const SmallTextButton({

    Key? key,

    required this.text,

    required this.ratio,

    required this.onPreesed,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return TextButton(

        onPressed: onPreesed,

        style: TextButton.styleFrom(

            primary: Colors.white,

            textStyle: TextStyle(

              fontSize: 18 \* ratio,

              fontFamily: 'Montserrat',

              fontWeight: FontWeight.w600,

            )),

        child: Text(text));

  }

}

class StyledButton extends StatelessWidget {

  final String text;

  final double ratio;

  final VoidCallback onPressed;

  const StyledButton({

    Key? key,

    required this.ratio,

    required this.text,

    required this.onPressed,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return ElevatedButton(

      onPressed: onPressed,

      child: Text(text,

          style: TextStyle(

              fontFamily: 'Montserrat',

              fontSize: 24 \* ratio,

              fontWeight: FontWeight.w700)),

      style: ElevatedButton.styleFrom(

        primary: buttonColor,

        fixedSize: Size.fromHeight(56 \* ratio),

        shape: RoundedRectangleBorder(

          borderRadius: BorderRadius.all(

            Radius.circular(20 \* ratio),

          ),

        ),

      ),

    );

  }

}

class StyledIconButton extends StatelessWidget {

  final Icon icon;

  final Text text;

  final double ratio;

  const StyledIconButton(

      {Key? key, required this.icon, required this.ratio, required this.text})

      : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Expanded(

      child: TextButton.icon(

          label: text,

          icon: icon,

          style: TextButton.styleFrom(

            padding: EdgeInsets.zero,

            fixedSize: Size.fromHeight(50 \* ratio),

            backgroundColor: fieldColor,

            shape: RoundedRectangleBorder(

                borderRadius: BorderRadius.all(

                  Radius.circular(20 \* ratio),

                ),

                side: const BorderSide(

                    color: Color.fromRGBO(238, 243, 245, 1), width: 0.2)),

          ),

          onPressed: () {}),

    );

  }

}

### menu.dart

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

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:song\_feeler/resources/socket.dart';

class StlyedIcon extends StatelessWidget {

  final double widthRatio;

  final double heightRatio;

  final Function onPressed;

  final MSocket socket;

  const StlyedIcon({

    Key? key,

    required this.heightRatio,

    required this.widthRatio,

    required this.onPressed,

    required this.socket,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Align(

      alignment: Alignment.center,

      child: Container(

        height: 360 \* heightRatio,

        width: 260 \* widthRatio,

        decoration: BoxDecoration(

            color: const Color.fromRGBO(59, 60, 80, 1),

            borderRadius: BorderRadius.all(Radius.circular(20.0 \* heightRatio)),

            boxShadow: [

              BoxShadow(

                color: const Color.fromRGBO(244, 231, 255, 1).withOpacity(0.1),

                spreadRadius: 10,

                blurRadius: 20,

              ),

            ]),

        child: Padding(

          padding: EdgeInsets.only(left: 12 \* widthRatio),

          child: TextButton.icon(

              onPressed: () {

                onPressed(socket);

              },

              icon: ImageIcon(

                const AssetImage('assets/3.png'),

                size: 220 \* heightRatio,

                color: const Color.fromRGBO(228, 195, 255, 1),

              ),

              style:

                  TextButton.styleFrom(splashFactory: NoSplash.splashFactory),

              label: const Text('')),

        ),

      ),

    );

  }

}

class StyledImage extends StatelessWidget {

  final double widthRatio;

  final double heightRatio;

  final File? image;

  const StyledImage({

    Key? key,

    required this.heightRatio,

    required this.widthRatio,

    required this.image,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Align(

      alignment: Alignment.center,

      child: Container(

        height: 360 \* heightRatio,

        width: 260 \* widthRatio,

        decoration: BoxDecoration(

            image: DecorationImage(

              fit: BoxFit.fill,

              image: FileImage(image!),

            ),

            color: const Color.fromRGBO(59, 60, 80, 1),

            borderRadius: BorderRadius.all(Radius.circular(20.0 \* heightRatio)),

            boxShadow: [

              BoxShadow(

                color: const Color.fromRGBO(244, 231, 255, 1).withOpacity(0.1),

                spreadRadius: 10,

                blurRadius: 20,

              ),

            ]),

      ),

    );

  }

}

class SmallText extends StatelessWidget {

  final double ratio;

  const SmallText({

    Key? key,

    required this.ratio,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Text(

      'Upload your face and feel your music.',

      style: TextStyle(

          fontSize: 17 \* ratio,

          fontFamily: 'Montserrat',

          fontWeight: FontWeight.w500,

          color: const Color.fromRGBO(244, 231, 255, 1)),

    );

  }

}

class BigText extends StatelessWidget {

  final double ratio;

  const BigText({

    Key? key,

    required this.ratio,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Text(

      'Hey You!',

      style: TextStyle(

          fontSize: 42 \* ratio,

          fontFamily: 'Montserrat',

          fontWeight: FontWeight.w800,

          letterSpacing: 1.1,

          color: const Color.fromRGBO(244, 231, 255, 1)),

    );

  }

}

class StyledTextButton extends StatelessWidget {

  final double heightRatio;

  final double widthRatio;

  final Function onPressed;

  final MSocket socket;

  const StyledTextButton({

    Key? key,

    required this.heightRatio,

    required this.widthRatio,

    required this.onPressed,

    required this.socket,

  }) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return Align(

      alignment: Alignment.center,

      child: TextButton(

        onPressed: () => onPressed(socket),

        style: ElevatedButton.styleFrom(

            padding: EdgeInsets.zero,

            shape: RoundedRectangleBorder(

                borderRadius: BorderRadius.circular(20 \* heightRatio))),

        child: Ink(

          decoration: BoxDecoration(

              gradient: const LinearGradient(colors: [

                Color.fromRGBO(118, 176, 253, 1),

                Color.fromRGBO(147, 143, 249, 1)

              ]),

              borderRadius: BorderRadius.circular(20 \* heightRatio)),

          child: Container(

              width: 310 \* widthRatio,

              height: 70 \* heightRatio,

              alignment: Alignment.center,

              child: Text(

                'Feel Your Music',

                style: TextStyle(

                    fontSize: 32 \* heightRatio,

                    fontFamily: 'Montserrat',

                    fontWeight: FontWeight.w800,

                    letterSpacing: 1.1,

                    color: const Color.fromRGBO(244, 231, 255, 1)),

              )),

        ),

      ),

    );

  }

}

class StyledAppBar extends StatelessWidget {

  final double ratio;

  final String page;

  final MSocket socket;

  const StyledAppBar({

    Key? key,

    this.fabLocation = FloatingActionButtonLocation.endDocked,

    required this.ratio,

    required this.page,

    required this.socket,

  }) : super(key: key);

  final FloatingActionButtonLocation fabLocation;

  @override

  Widget build(BuildContext context) {

    return BottomAppBar(

      color: const Color.fromRGBO(19, 19, 30, 1),

      child: IconTheme(

        data: IconThemeData(color: Theme.of(context).colorScheme.onPrimary),

        child: Row(

          mainAxisAlignment: MainAxisAlignment.spaceEvenly,

          children: <Widget>[

            IconButton(

              tooltip: 'Home',

              iconSize: 40 \* ratio,

              icon: const Icon(Icons.home\_rounded),

              onPressed: () => MSocket.pushReplacement(context, page, socket),

            ),

            IconButton(

              tooltip: 'History',

              iconSize: 40 \* ratio,

              icon: const Icon(Icons.history\_rounded),

              onPressed: () {},

            ),

            IconButton(

              tooltip: 'Profile',

              iconSize: 40 \* ratio,

              icon: const Icon(Icons.person\_rounded),

              onPressed: () {},

            ),

          ],

        ),

      ),

    );

  }

}

### socket.dart

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

import 'dart:io';

import 'dart:convert';

import 'dart:async';

import "dart:typed\_data";

import 'package:flutter/material.dart';

class MSocket {

  Completer completer = Completer();

  late Socket socket;

  late StreamSubscription<Uint8List> stream;

  final TextEditingController \_textFieldController = TextEditingController();

  MSocket();

  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";

  }

  void close() {

    try {

      sendMessage("", 4);

      socket.close();

      // ignore: empty\_catches

    } catch (e) {}

  }

  Future<String> sendAndWaitForImage(

      BuildContext context, Uint8List message, int code) async {

    try {

      completer = Completer();

      sendImage(context, message, code);

      String reply = await receiveMessage();

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

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

        return "2";

      }

      return reply;

    } on SocketException {

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

    }

    return "2";

  }

  String extractData(Uint8List data) {

    int start = data.indexWhere((element) => element == 49);

    int length = ByteData.view(data.sublist(start + 1, start + 5).buffer)

        .getInt32(0, Endian.little);

    if (start + 5 == start + 4 + length) {

      return String.fromCharCode(data.elementAt(start + 5));

    } else {

      return String.fromCharCodes(data.sublist(start + 5, start + length));

    }

  }

  static Uint8List \_int32bytes(int value) =>

      Uint8List(4)..buffer.asInt32List()[0] = value;

  void sendMessage(String message, int code) {

    List<int> bytes = [code];

    bytes.addAll(\_int32bytes(message.length));

    bytes.addAll(utf8.encode(message));

    socket.add(bytes);

  }

  void sendImage(BuildContext context, Uint8List message, int code) {

    try {

      List<int> bytes = [code];

      bytes.addAll(\_int32bytes(message.length));

      bytes.addAll(message);

      socket.add(bytes);

    } on SocketException {

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

    }

  }

  Future<void> \_displayTextInputDialog(BuildContext context) async {

    return showDialog(

      context: context,

      barrierDismissible: false,

      builder: (context) {

        return AlertDialog(

          title: const Text('Server IP'),

          content: TextField(

            keyboardType: TextInputType.number,

            controller: \_textFieldController,

            decoration: const InputDecoration(hintText: "0.0.0.0"),

          ),

          actions: <Widget>[

            TextButton(

              child: const Text('OK'),

              onPressed: () {

                Navigator.pop(context);

              },

            ),

          ],

        );

      },

    );

  }

  static Future<void> showError(

      BuildContext context, String title, String content) async {

    return showDialog(

      context: context,

      barrierDismissible: false,

      builder: (context) {

        return AlertDialog(

          title: Text(title),

          content: Text(content),

          actions: <Widget>[

            TextButton(

              child: const Text('OK'),

              onPressed: () {

                Navigator.pop(context);

              },

            ),

          ],

        );

      },

    );

  }

  static void pushReplacement(

      BuildContext context, String page, MSocket socket) {

    Navigator.pushReplacementNamed(context, page,

        arguments: {"Socket": socket});

  }

  static void pushReplacementWithArgs(

      BuildContext context, String page, Map<String, dynamic> object) {

    Navigator.pushReplacementNamed(context, page, arguments: object);

  }

}

### קוד המכונה

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras.layers.core import Flatten, Dense, Dropout

from keras.layers.convolutional import Convolution2D, MaxPooling2D, ZeroPadding2D

from keras.optimizers import gradient\_descent\_v2

import cv2

!wget https://www.dropbox.com/s/va90dnmndh0x8el/28577\_36420\_bundle\_archive.zip

!unzip 28577\_36420\_bundle\_archive.zip

x\_train, y\_train, x\_test, y\_test = [], [], [], []

df = pd.read\_csv('/content/fer2013.csv')

for index, row in df.iterrows():

    val = row['pixels'].split(" ")

    try:

      if 'Training' in row['Usage']:

          x\_train.append(np.array(val, 'float32'))

          y\_train.append(row['emotion'])

      elif 'PublicTest' in row['Usage']:

          x\_test.append(np.array(val, 'float32'))

          y\_test.append(row['emotion'])

    except:

      print("Error!!! at index - {} and row - {} ".format(index, row))

num\_features = 64

num\_labels = 7

batch\_size = 64

epochs = 50

width, height = 48, 48

x\_train = np.array(x\_train, 'float32')

y\_train = np.array(y\_train, 'float32')

x\_test = np.array(x\_test, 'float32')

y\_test = np.array(y\_test, 'float32')

from keras.utils.np\_utils import to\_categorical

y\_train = to\_categorical(y\_train, num\_classes=num\_labels)

y\_test = to\_categorical(y\_test, num\_classes=num\_labels)

  # normalizing our data 0 and 1

  x\_train -= np.mean(x\_train, axis=0)

  x\_train /= np.std(x\_train, axis=0)

  x\_test -= np.mean(x\_test, axis=0)

  x\_test /= np.std(x\_test, axis=0)

  x\_train = x\_train.reshape(x\_train.shape[0], 48, 48, 1)

  x\_test = x\_test.reshape(x\_test.shape[0], 48, 48, 1)

from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D, Activation

from tensorflow.keras.preprocessing import image

import matplotlib.pyplot as plt

from tensorflow.keras.losses import categorical\_crossentropy

from tensorflow.keras import layers

from tensorflow import keras

from tensorflow.keras.layers import BatchNormalization

model = Sequential()

# 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.

## 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>