IdanMusic

Name: Idan Boros
ID:323837310
School: Makif Chet
Teacher: Jacob Shutzman
Date: 20.6.2020

תוכן עניינים

מבוא 3
מבנה הפרויקט + מדריך למפתח (מצורפים גם הסברים אודות משתנים ופונקציות): 4
פונקציה להתחברות: 5
פונקציה ליצירת משתמש 6
הסבר אודות מודול ההתחברות: 8
שרת REST 9
DFD אודות העלאת שירים לשרת REST 11
Client GUI and data structures (Classes file) 12
Classes 12
מחלקת SongFile 13
מחלקת GuiSong 13
מחלקות Playlist, Queue 14
מחלקות מסוג Empty 14
מחלקת user 14
פונקציות songfilefromDB, youtubeid_toplay, one_search, make_image, unshorten_url, make_tiny 14
הסבר אודות מחלקת הGUI 15
הסבר אודות threadedclasses 18
ThreadedYTSearch 19
ThreadingDisplay 19
ThreadedSongSearch 20
ThreadedDisplayPlaylist 21
פונקצית from_crs_to_song 22
תרשים UML יותר מורחב של כל העצמים בפרויקט 23
מדריך למשתמש – נלקח מתוך readme.md 24
Signing up and getting set up: 24
Signing Up! 24
Logging in! 24
Navigating the app! 25
Searching AND Playing 25
Playlists 25
Notes: 26
הסבר בסיס הנתונים 27
songDB 27
usrDB 28
רפלקציה 29
נספחים 30
פיצ'רים עתידיים שאשמח להוסיף 30
ביבליוגרפיה 31

מבוא

הספר יכלול את המבנה של הפרוייקט שלי והסברים בנוגע לארכיטקטורה שלו, מדריך למשתמש, הסבר בנוגע לבסיס הנתונים שהשתמשתי בו והמבנה שלו (NoSQL במקרה שלי) ומדריך למפתח, עם הסברים בנוגע לקוד ומדוע נכתב בצורה שנכתב (בנוסף להערות על קטעי הקוד עצמם).

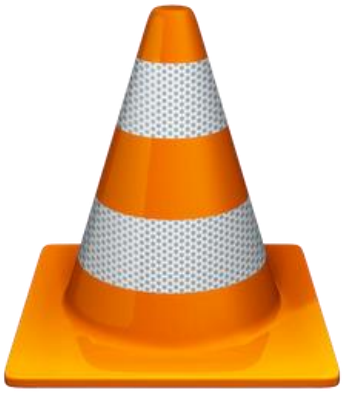
אני נורא אוהב לשמוע מוזיקה, ואולי זה ישמע מוזר לרוב האנשים אבל האיכות של המוזיקה משנה לי, אפילו אם ההבדל אינו ניכר. לשמוע מוזיקה שאני ממש אוהב אותה באיכות של 320 kbps או בקובץ שלא עבר דחיסה ונלקח היישר מדיסק או מתקליט זה משנה לי. ובעקבות כך שאין אפשרויות לסטרימינג באיכות גבוהה ברוב השירותים כגון יוטיוב וספוטיפיי החלטתי להכין פתרון משל עצמי, בשם IdanMusic, החלטתי להכין בעצם שירות סטרימינג לשירים שמיועד לכך שיהיו בו שירים באיכות גבוהה, לאחר תחילת הפיתוח הבנתי שעל מנת שמאגר השירים יהיה רחב לא אוכל להסתמך על העלאות שלי בלבד אליו והחלטתי לצרף עוד מקור למאגר: יוטיוב, נכון, זה די מבטל את כל העניין של הזרמת שירים באיכות גבוהה אבל התוכנה יודעת לבחור את האופציה הכי טובה מבחינת איכות סאונד שנמצאת ביוטיוב באמצעות שימוש בAPI שמממש את הAPI של יוטיוב בפייתון בשם Pafy. וכמו כן, אפשר להבדיל בתוכנה בין שירים שנקלחו מיוטיוב לבין שירים שהגיעו מהDB, שכנראה יהיו באיכות גבוהה יותר.

שני החידושים העיקריים בפרויקט הם ראשית : האופציה לשיתוף שירים באיכות גבוהה (פורמט FLAC), לצד לקיחת שירים מיוטיוב.
שנית, עצם העובדה שלמשתמשים יש אופציה להעלות שירים משלהם למאגר שיהיו חשופים לכל העולם, ולהעלות באיכות גבוהה בפרט. ההעלאה מתבצעת לשרת REST, הhost שבו בחרתי הוא PythonAnyWhere

השירים אשר מקורם בDB ובאיכות גבוהה (קבצי lossless) יהיו ניתנים להורדה בתשלום (נכון לעכשיו הכל חינם כי אין מערכת כזו בתוכנה, אך ניתן לממש אחת), וההורדה שלהם תקח יותר זמן מהזמן שלוקח לסטרימינג של שיר רגיל בפורמט mp3.

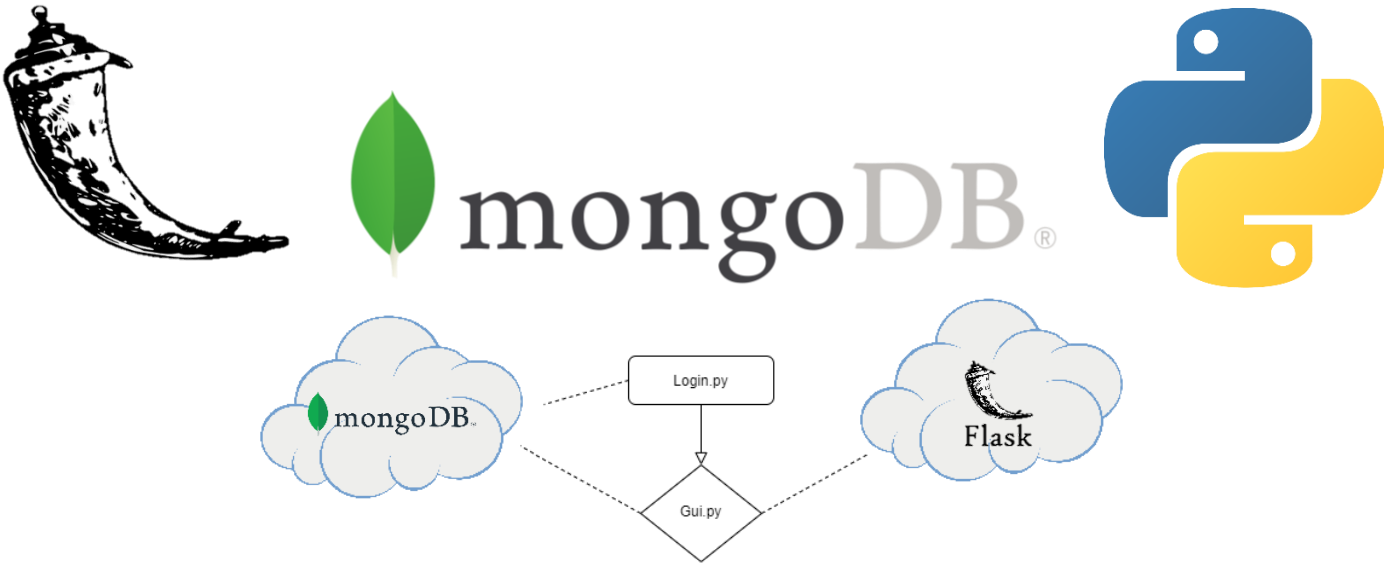
בפרויקט נתקלתי בכמה אתגרים, ראשית, איך אוכל להביא פרטים על שירים מיוטיוב, בזמן שביוטיוב כל אחד יכול להעלות את השיר בלי שום MetaData? (מידע על השיר שאגור בקובץ השיר כגון תמונת אלבום, אלבום, אמן, שנת הוצאה ועוד) יתר על כן, איפה אוכל למקם את קובץ השרת בכדי שהוא יהיה נגיש מכל מחשב בעולם? איך אוכל לשמור על הסיסמאות של המשתמשים באופן בטוח? איך אפשר להפוך קישור מיוטיוב ל"זרם" (סטרים) של מוזיקה, ואחרי שעשיתי את זה איך אפשר לנגן אותו? לכל הבעיות הללו מצאתי פתרון.

הפתרונות לבעיות העיקריות שנחשפתי אליהן בפרויקט כללו מחקר של הנושא הבעייתי, לדוגמה כשהיה לי קישור לסטרים של יוטיוב ולא ידעתי איך לנגן אותו חיפשתי באינטרנט על אופציות ולבסוף הפתרון הטוב ביותר שמצאתי הוא מודול של VLC אשר קיים בפייתון ומאפשר לנגן סטרים כזה וגם לשלוט בווליום, הזמן, ועוד כל מיני פרמטרים.



מבנה הפרויקט + מדריך למפתח (מצורפים גם הסברים אודות משתנים ופונקציות):

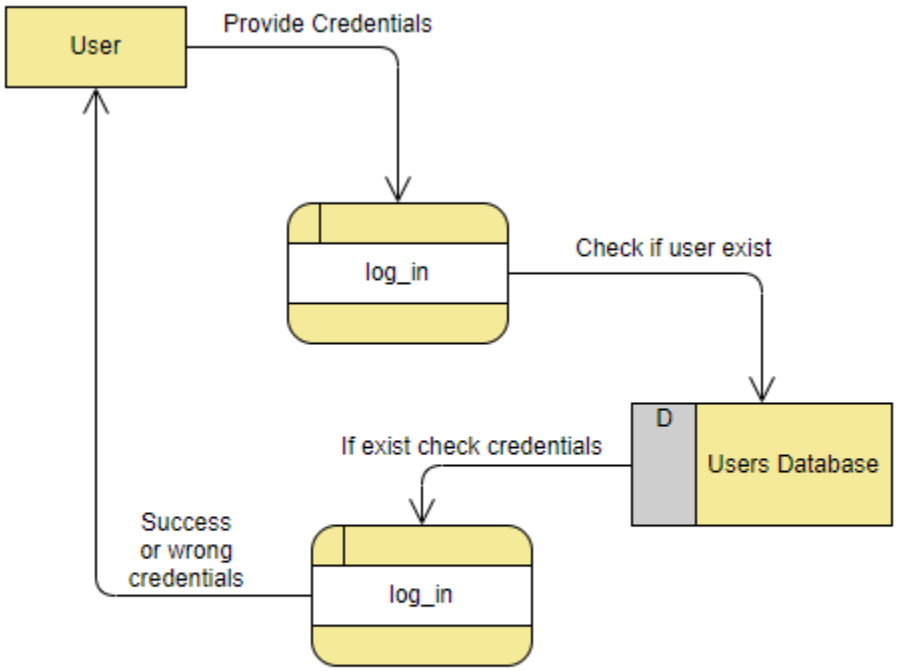
הפרויקט נכתב בשפת פייתון, הממשק הגרפי נכתב באמצעות Tkinter, והשרת נכתב באמצעות Flask, בפייתון.
השתמשתי בMongoDB על מנת לאחסן את הפרטים על השירים שהועלו למאגר ואת הפרטים על המשתמשים.



התוכנה מתחילה ממסך Login, שבו ניתן לבצע הרשמה לשירות או להכנס למשתמש קיים, הממשק בקובץ Login.py נכתב באמצעות Tkinter.

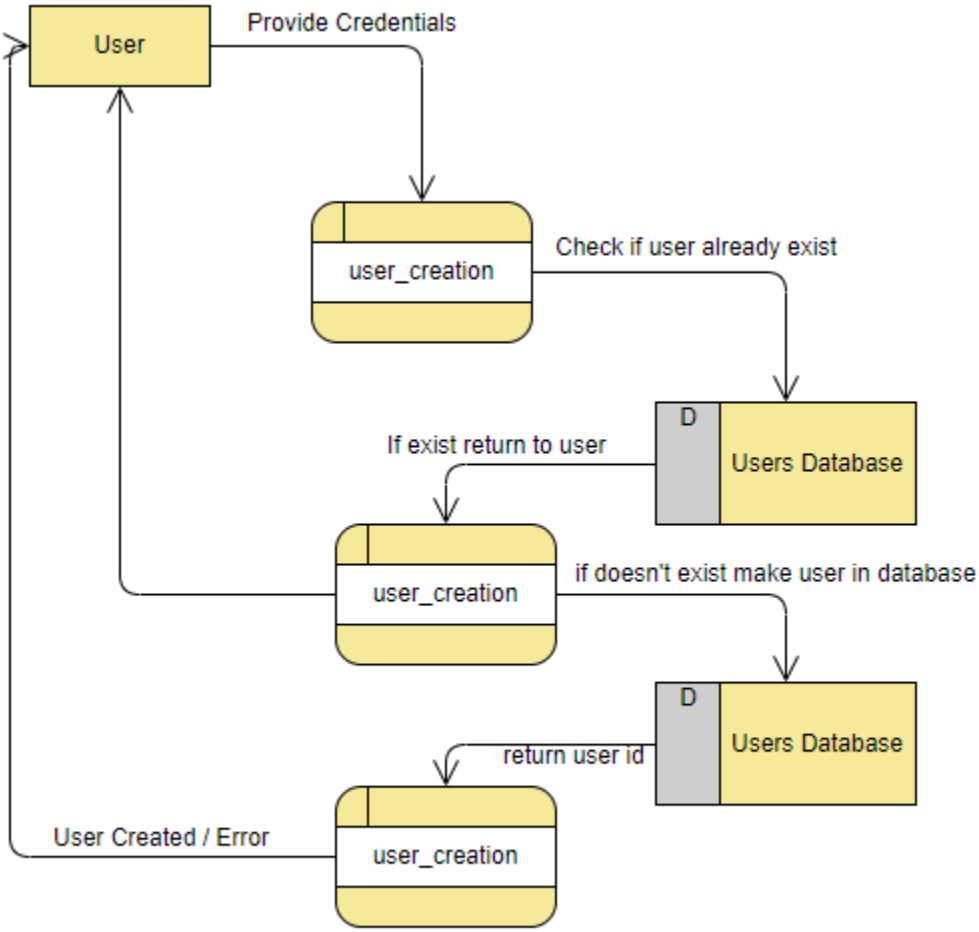
פונקציה להתחברות:

```
def log_in(event=None, db_crt=None):
    """Function that logs the user into the program and starts it up."""
    if event is None:
        pass
    crtuser = User(userentry.get(), pass_entry.get())
    cloud_user = db_crt.usrdb.find({'username': crtuser.username})[0]
    print(str(cloud_user))
    print(str(crtuser))
    strng_decode = sha512()
    strng_decode.update(crtuser.password.encode('utf-8'))
    print(strng_decode.hexdigest())
    if strng_decode.hexdigest() == cloud_user['password'] \
            and crtuser.username == cloud_user['username']:
        messagebox.showinfo("Logged in!", "Congrats")
        print(f"User : {crtuser.username}, Password :", str(crtuser.password))
        argv.append(crtuser.username)
        argv.append(crtuser.password)
        import gui
        for widget in root.winfo_children():
            widget.destroy()
        gui.AppWindow(root, user=crtuser.username, password=crtuser.password, db=db_crt)
    else:
        messagebox.showwarning("Credentials problem!", "wrong user or password")
```



פונקציה ליצירת משתמש

```
def user_creation(db_crt=None):
    """Function that creates a user in the database."""
    usrdb = db_crt.usrdb
    search = usrdb.find({'username': userentry.get()})
    bool1 = False
    for post in search:
        print("Look! already exists:"+str(post))
        bool1 = True
    if bool1:
        messagebox.showwarning("Credentials problem!", "Username taken!")
        return
    if ' ' in userentry.get() \
            or userentry.get() == '' or len(userentry.get()) <= 4:
        messagebox.showwarning("Credentials problem!", "Bad username!")
        return
    if not check_pass(pass_entry.get()):
        messagebox.showwarning("Credentials problem!", "Password does not meet criteria\n"
                               "(6 letters, 3 individual "
                               "letters and does not include a space)")
        return
    string_encode = sha512()
    string_encode.update(str(pass_entry.get()).encode('utf-8'))
    answer = simpledialog.askinteger("Preferred search view",
                             "How many songs would you like to be "
                             "showed to you when you search for songs?",
                             parent=root)
    if answer:
        crtuser = User(userentry.get(), str(string_encode.hexdigest()), answer)
        print('made user1, ', crtuser)
    else:
        crtuser = User(userentry.get(), str(string_encode.hexdigest()))
        print('made user2, ', crtuser)
    crtuser = crtuser.to_json()
    usr_id = usrdb.insert_one(crtuser).inserted_id
    print(crtuser, ' on id ', usr_id)
```


User
Provide Credentials
user_creation
Check if user already exist
D
Users Database
If exist return to user
user_creation
if doesn't exist make user in database
D
Users Database
return user id
user_creation
User Created / Error

הסבר אודות מודול ההתחברות:

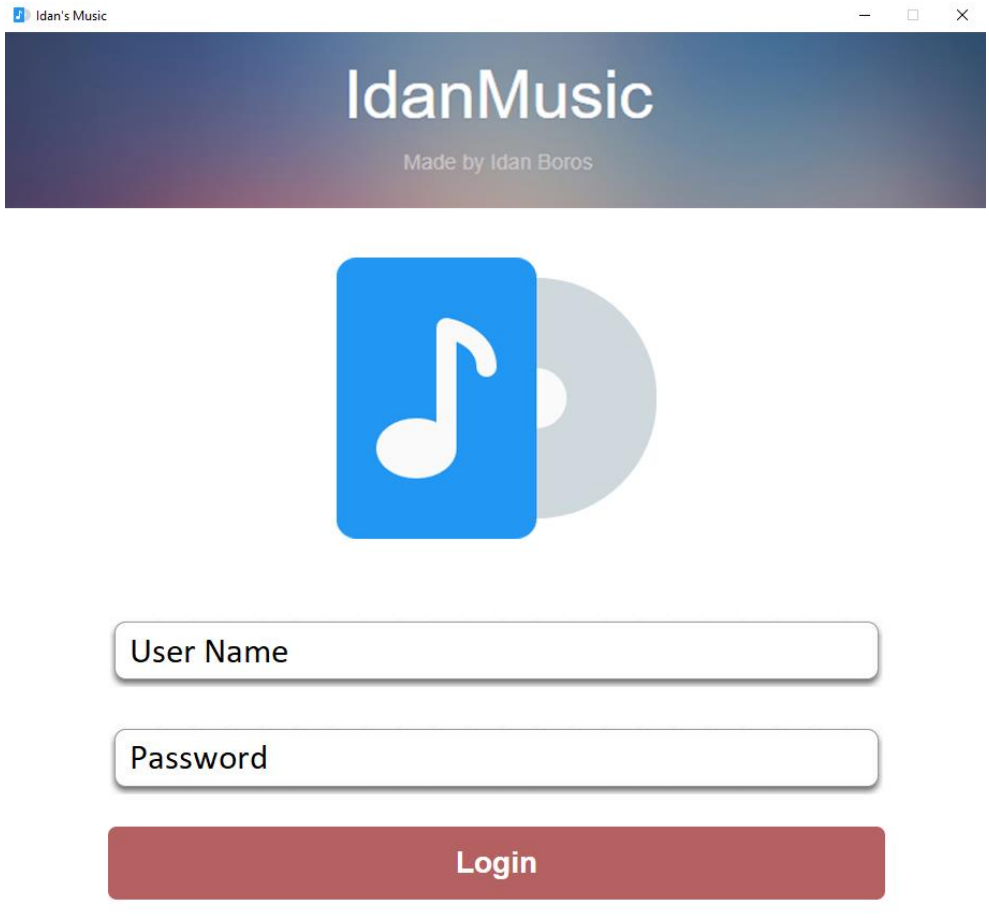
יחידת(מודול) ההתחברות מקבלת 5 קלטים סך הכל: ראשית, 2 קלטים מן המשתמש, שם משתמש וסיסמא, ובמקרה של יצירת משתמש גם מספר השירים שיעדיף לראות כאשר יבצע חיפוש.
שנית, קלטים ממאגר הנתונים: האם המשתמש קיים? ואם מנסים להתחבר, את הפרטים אודות המשתמש אליו מנסים לגשת.

יחידת ההתחברות אינה מחזירה פלט פרט לפלט בסיסי מאוד של "התחברות התקבלה" או "שגיאה בפרטים", מכיוון שאין צורך ביותר.

היחידה מבצעת שימוש במסד נתונים של MongoDB, במקרה השימוש שלי המסד מאוחסן ברשת דרך Mongo, אך ניתן גם להפוך אותו למאגר לוקאלי באמצעות שינוי הערך של מסד הנתונים להתחברות למסד לוקאלי.

היחידה מבצעת גיבוב בפורמט SHA512 לסיסמאות המשתמשים על מנת לוודא כי מסד הנתונים מאובטח והסיסמאות לא חשופות לגניבה, בשביל לוודא זאת היא עושה את זה באופן לוקאלי על המחשב לפני שליחת הסיסמה בכדי שהסיסמה לא תהיה גלויה בplaintext בפקטה שנשלחה ברשת באמצעות תוכנות Sniffing כדוגמת Wireshark.

פיצ'ר עתידי שאשמח להוסיף יהיה האופציה לשנות סיסמה או שם משתמש.



2.הממשק הגרפי של מודול ההתחברות

שרת REST

בפרויקט קיים כמו כן גם שרת REST שמאוחסן בשרתי PytohnAnywhere, השרת נועד להעלאת שירים ולניגונם על ידי המשתמשים, בעתיד תהיה אופציה לצרף תשלום לשירים באיכות גבוהה (קבצים לא דחוסים), אך כעת הכל בחינם.

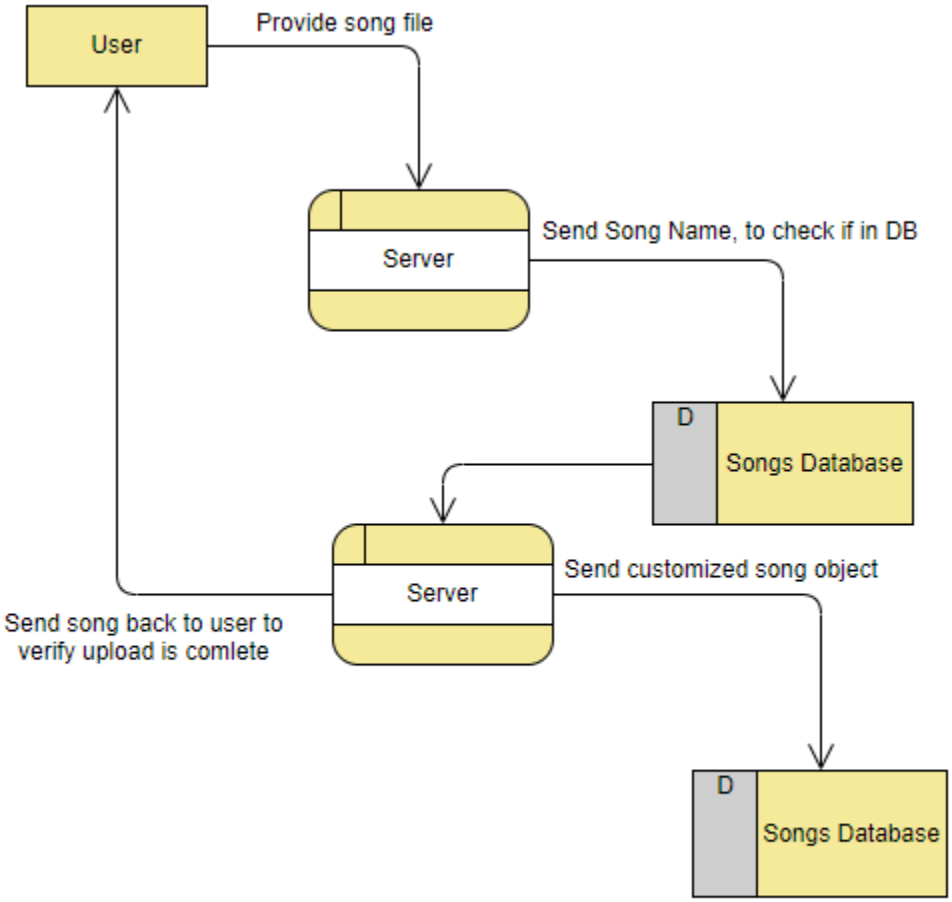
השרת מאוחסן בכתובת https://idanbb.pythonanywhere.com

קטע הקוד אשר אחראי על העלאת השירים לשרת הREST.

```
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            print("THE OS CWD IS : ")
            print("TYPE OF FILE IS : ", type(file))
            # Support for hebrew
            if detect(file.filename) == 'en':
                filename = secure_filename(file.filename).lower().encode()
            else:
                try:
                    filename = secure_filename(file.filename).encode()
                except:
                    filename = file.filename.encode()
                finally:
                    print("Fname is ", str(filename.decode()))
            print("THE OS CWD IS 2: ", 'songname is ', file.filename)
            open((os.path.join(app.config['UPLOAD_FOLDER'], str(filename.decode()))), 'x')
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], str(filename.decode())))
            # The server upload is done here, from here on out is interaction with db
            with open(os.path.join(app.config['UPLOAD_FOLDER'], str(filename.decode()))) as fp:
                # Makes an appropriate song object for the database to be appended.
                s = SongFile(fp, name=os.path.splitext(file.filename)[0])
                print("Artist is", s.artist)
                if s.artist == "Unknown":
                    # The next part is using spotify's API to generate a song object,
                    # This part will not be included because it's long and irrelavent.
                if " " in songname:
                    #This is to make sure the song doesn't
                    #have any spaces, so it can be indexed in the server.
                    songname = songname.replace(' ', '_')
                    print("After fix is:", songname)
                # only adds the song to the db if it isn't there yet.
```

```
            if not db.songdb.find_one({'name': s.name}):
                s.filename = songname
                s.streamurl = s.streamurl+s.filename
                if s.prefix == "":
                    s.prefix == '.mp3'
                db.songdb.insert_one(s.to_json())
            print(songname)
            fp.close()
        try:
            """ Tries to rename the song name to the format accepted in the server,
            if it doesn't work it means the song was already uploaded before and it
            shouldn't accept a repuload so it doesn't upload the song."""
            os.rename(os.path.join(app.config['UPLOAD_FOLDER'], str(filename.decode())),
os.path.join(app.config['UPLOAD_FOLDER'],songname) + s.prefix)
        except FileExistsError as e:
            print("Current uploaded song existed")
        except Exception as e:
            exc_type, exc_obj, exc_tb = sys.exc_info()
            print(exc_type, exc_tb.tb_lineno, e)
        return redirect(url_for('uploaded_file', filename=songname+s.prefix))
    else:
        return '''
        <title>Sorry, file format not supported!</title>
        <h1>Upload new File</h1>
        <form method=post enctype=multipart/form-data>
          <input type=file name=file>
          <input type=submit value=Upload>
        </form>
        '''
  return '''
  <!doctype html>
  <title>Upload new File</title>
  <h1>Upload new File</h1>
  <form method=post enctype=multipart/form-data>
    <input type=file name=file>
    <input type=submit value=Upload>
  </form>
  '''
```

DFD אודות העלאת שירים לשרת REST



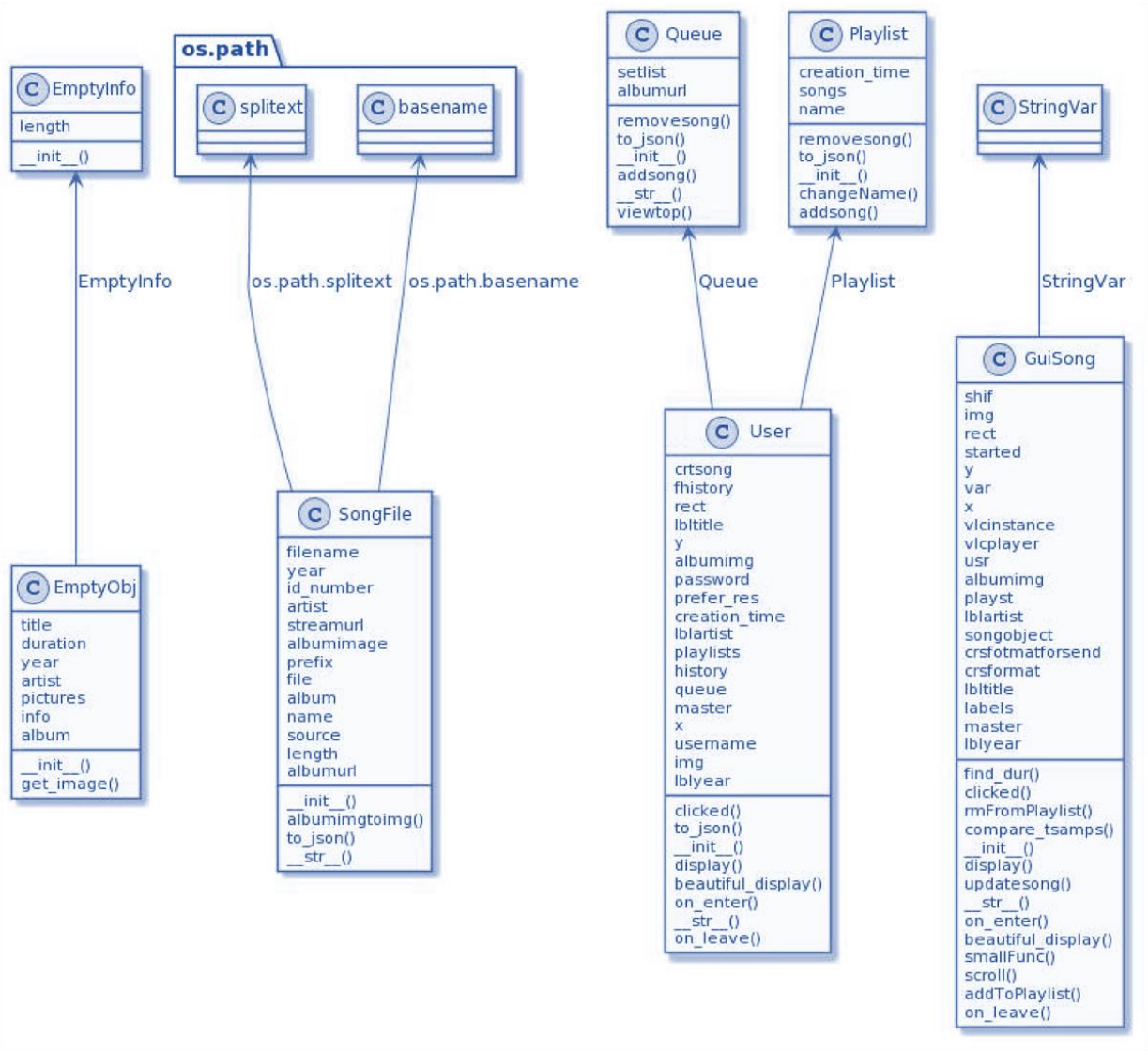
Client GUI and data structures (Classes file)

בחלק הזה אסביר על ממשק המשתמש, איך שהוא בנוי ועל הקובץ שמכיל את העצמים בפרויקט ואחראי על מבני הנתונים (כגון משתמש, שיר, פלייליסט וכו'.), יתכן ואשמיט הרבה קוד בחלק הזה כי אלה דברים פשוטין שאני לא מרגיש צורך לפרט עליהם בקוד ואני מאמין שהתרשים יספיק (כגון הדרך שבה מיוצר קובץ שיר, קוד נורא ארוך שפשוט מתאים לכל פורמט שיר ודואג שהכל יעבוד כמו שצריך).

Classes

ראשית, אסביר את הקובץ אשר מכיל את הקלאסים(מחלקות) בפרויקט.

הקובץ מכיל פשוט מחלקות שיש שימוש בהן בפרויקט, אסביר אודות כל מחלקה בנפרד.



מחלקת SongFile

המחלקה הזו אחראית על העצמים אשר מכילים את הנתונים אודות שיר, משם הקובץ שלו ועד לאורך שלו, ליתר דיוק היא מכילה את הפרטים הבאים:

שם, שם אמן, מקור (מאגר נתונים או יוטיוב?), כתובת של תמונת אלבום (לא חייב לקבל פרט זה, קיימת תמונה דיפולטיבים למקרה שלא מקבלים), כתובת סטרימינג של השיר, שם אלבום, סיומת (של קובץ השיר, האם השיר ניתן לסטרימיניג?), filename – משמע שם הקובץ עצמו, קיים פרמטר גם בשם "קובץ" שמכיל path לקובץ אבל הפרמטר פיילניים מכיל רק את השם של הקובץ. פרמטר של תמונת אלבום (אם השיר נמצא במאגר ולא מגיע מיוטיוב הוא מקבל משם את התמונה שלו), אורך שיר, מספר מזהה אם השיר הגיע מיוטיוב, ושנה שבה השיר יצא.

כל הפרטים האלה הכרחיים בכדי לגרום למנגון לעבוד (טוב בכנות, אפשר לוותר על פרמטר האורך אבל המערכת בנויה באופן שבו הוא נחוץ!)

המחלקה בשימוש במגוון מקומות, ואחד מבין החשובים מהם הינו המחלקה GuiSong. וגם מאגר הנתונים עליו ארחיב בהמשך.

מחלקת GuiSong

מחלקה זו אחראית על התצוגה של השירים במערכת, העצמים במחלקה זו מקבלים שלל פרמטרים.

master – הפריים שבו השיר יוצג.
songobject – העצם שדואג לפרטי השיר שיש להציג.
x,y – המיקום שבו יש להציג את השיר על פני הפריים.
vlcplayer, vlcinstance – הפרטים אודות הנגן שמבקש לנגן את השיר.
usr – המשתמש שביקש לנגן את השיר.
started – בודק האם הנגן שביקש לנגן את השיר ניגן שיר אחר לפני כן (האם התחילו לנגן איתו דברים) כי אם עוד לא ניגנו איתו דברים יש לשנות דבר נוסף.
playst – הפלייליסט שאליו השיר משתייך.

במחלקה מוגדרות מגוון מתודות פנימיות שנותרו לטפל בכל מיני מקרים.

smallFunc – מתודה שנועדה להקרא כאשר מאשרים לאיזה פלייליסט מעוניינים להוסיף שיר, ומוסיפה את השיר לפלייליסט, שמע נובע מהעובדה שאורכה 2 שורות.
rmFromPlaylist – מתודה שמטרתה להסיר את השיר מהפלייליסט אליו משתייך במקרה ונקראה.
addToPlaylist – מתודה שמטרתה להוסיף את השיר לפלייסט מבוקש.
on_enter, **on_leave** – מתודות שנועדו להקרא כאשר הסמן מצביע מעל השיר, מכיוון שאז הרקע של השיר משנה את צבעו.
clicked – מתודה שנקראת כאשר לוחצים על שיר ורוצים לנגנו.
compare_tsamps – מתודה שמשווה בין 2 "נקודות זמן" במקרה והשיר מקורו ביוטיוב, מכיוון שקישורי יוטיוב עלולים להיות פגי תוקף לאחר כמה זמן, גיליתי זו בשימוש ממושך בתוכנה שקישורים כלשהם פשוט הפסיקו לעבוד לאחר כמה זמן למרות שנשמרו בפלייליט.
find_dur – מתודה שמחלצת את אורכו של השיר מURL של יוטיוב מכיוון שהאורך "מסתתר" שם.
scroll – מותדה שאחראית במקרה שהשיר ארוך מידי לגלול אותו בכדי שיכנס במסך.
display – מתודה שאחראית על הצגת השיר בפריים הנתון.
beautiful_display – מתודה שאחראית על הצגה יפה של שיר (הצגה גדולה).
updatesong – מתודה שאחראית להציג את התמונה של השיר מחדש, במקרה וקרתה בעיה.

מחלקות Playlist, Queue

שתי המחלקות הנ"ל הן מבני נתונים של משתמשים, אחד מייצג רשימת שירים והשני מייצג תור של שירים, לשניהם רשימת שירים בתור אחד המאפיינים, לפלייליסט מאפיין של שם מכיוון שהוא צריך שם ולשניהם יש מתודות שמאפיינות אותם בתור מבני נתונים (לתור קיימת מתודת viewtop וpop ולפלייסט מתודה להוספת שירים, החלפת שם והסרת שירים).

מחלקות מסוג Empty

מחלקות אלו נועדו לדמות אובייקטים ריקים למקרה שבו לא מתנגן שיר או שקיימת בעיה בנתוני השיר אז יש להציג שיר גנרי שמצביע על תקלה, מחלקות אלו דלות בתוכן וחסרות מתודות.

מחלקת user

אסביר אודות מחלקה זו בצורה מורחבת בהסבר על מבנה הנתונים מכיוון ששם זה יותר רלוונטי עקב כך שמחלקה זו נשמרת במבנה הנתונים ומיושמת שם המון.

פונקציות songfilefromDB, youtubeid_toplay, one_search, make_image, make_tiny, unshorten_url

לא ארחיב אודות הפונקציות הללו מכיוון שהן ארוכות ובסיסיות מידי בכדי שאסביר עליהן, אך אסביר בקצרה אודות כל אחת.

make_image: תפקידה לייצר תמונה על קנבס שמקבלת.

youtubeid_toplay: פונקציה שתפקידה להפוך id של סרטון ביוטיוב ללינק שאפשר לנגן, ולעשות לו סטרימינג. משתמשת בpafy.

one_search: פונקציה שתפקידה להפוך קישור יוטיוב בודד לאובייקט pafy ולבצע שאילתה בספוטיפיי בשביל לקבל מידע אודות השיר שאותה היא הופכת לאובייקט פאפי ולבסוף מחזירה אובייקט songfile שמתאר את השיר עם קישורית לסטרימינג שלו.

songfilefromDB: מקבלת אובייקט שהוצא ממאגר הנתונים בפורמט ג'ייסון והופכת אותו לאובייקט songfile.

make_tiny, unshorten_url: כאשר השירים מוזנים למאגר הנתונים המידע הקישוריות שלהם מקבלות קיצור בכדי לחסוך בגודל המאגר וגם שהפעולה תתרחש מהר יותר מכיוון שלמשוך קישורית גדולה ולדחוף קישורית גדולה יקח יותר זמן, לכן הפונקציות האלה נועדו לקצר את קישורית השיר ולהחזיר אותה לקדמותה בהתאם.

הסבר אודות מחלקת הGUI

על כלל הGUI אחראית מחלקה אחת שדואגת לכל מה שקורה שם, אסביר את הפונקציות שקיימות בה ואת שלל המשתנים.

מפאת אורכה ומורכבותה של המחלקה (כ1300 שורות) אני אנסה לתמצת חלק מהדברים ולהתמקד בחשובים. יתר על כן לא אוסיף קוד על מנת שכל ספר הפרויקט לא יהיה פשוט מלא בשורות קוד שאינן נחוצות, אלא רק בדברים רלוונטיים והסברים אודות הקוד, המטרתה שלשמה הספר נועד.

תחילה נתמקד בעצם בצורה שבה מוצג הGUI ואתאים כל רכיב בו למשתנה המייצג אותו:

AppWindow

albumimg : PhotoImage, NoneType
back_img : PhotoImage
backcont : Label
bigframe : Frame
bigframeobjects : dict
bigsongcanvas : Canvas, NoneType
bigsonglblalbum : Label
bigsonglblart : Label
bigsonglblartist : Label
bigsonglbltitle : Label
black_play : PhotoImage
crtalbumlabel : Label
crtartist_font : Font
crtartistlabel : Label
crtastyle : Style
crtbigalbum : Image, NoneType
crthour
crtplstbutton : Button
crtplstlabel : Label
crtsong_font : Font
crtsonglabel : Label
crtsongobj : SongFile
crtstyle : Style
crtuserbutton : Button
currentframe : Frame
db_mongo : NoneType
entryfontstyle : Font
grey_back : PhotoImage
grey_next : PhotoImage
grey_play : PhotoImage
gui_obj : NoneType, GuiSong
height : int
history_label : Label
img : NoneType, Image, PhotoImage
is_paused : bool
next_img : PhotoImage
nextcont : Label
on_big_song : bool
on_playlist : bool
password : str
pause_button : PhotoImage
playcont : Label
player : MediaPlayer
playlistdict : dict
playstframe : Frame
plsts : Label
removedplaylist : NoneType
root
scale_style : Style
search_lbl_src : Label
searchbox : Entry
seeking : bool
seekslidrstyke : Style
serverprefix : str
skipping : bool
slength : str, int, float
slidr : Scale
smallframe : Frame
smallimg : Image, NoneType
songcanv
songnamevar : StringVar
startout : list
times_playlists : Font
times_playlists_label : Font
user
usr : User, list
vlc_instance : Instance
volslidrstyle : Style
volumeslidr : Scale
welcome : Label
welcomefontstyle : Font
width : int

add_to_queue(playst)
check_time()
cleanFrame(event)
compare_tsamps(nextplaying)
configurewin(event)
create_big_frame(bigframe)
create_home(bigframe2)
delete_playlist(playst)
disable_textbox(event)
display_plsts()
enable_textbox(event)
find_dur(url)
load_first()
make_big_song_widget(song)
make_circle_image()
make_crsobject()
make_playlist(playst)
make_playlist_name(event)
on_enter_back(event)
on_enter_play(event)
on_enter_skip(event)
on_leave_back(event)
on_leave_play(event)
on_leave_skip(event)
on_press_back(event)
on_press_next(event)
on_press_play(event)
on_press_play_space(event)
play_next_song()
search_songs(event)
seeking_action(event)
shuffle_plst(plst)
slidr_seek(event)
slidr_volume_seek(event)
update_usr()
view_big_song(event)
view_search(event)

באמצעות התרשים הנ"ל ניתן להבין לאיזה עצם בממשק הגרפי מיוחס כל משתנה שרלוונטי לכך אלה הם בעצם העצמים והמשתנים העיקריים שיש שימוש בהם בחלון של הממשק הגרפי שלי. עכשיו אתחיל להסביר על המתודות הפנימיות המיושמות במחלקה זו (מופע שלה מזמן את החלון).

add_to_queue: מתודה שמטרתה להוסיף את כל השירים ברשימת השמעה מסויימת לתור השירים של המשתמש (כאשר לוחצים במקש ימני על רשימת השמעה כל שהיא היא מתווספת לתור של אותו המשתמש).

check_time: "מגה" מתודה אשר רצה ברקע כל הזמן ומוודאת שהכל בסדר, מקדמת את הסליידר בהתאם לשיר ודואגת שאם השיר נגמר יעבור לשיר הבא ושהתמונה תתחלף ועוד המון דברים. המתודה רצה כל 1 מילי שניות.

cleanframe: מתודה שנועדה לנקות את הפריים BigFrame ונקראת כאשר צריך לבצע זאת (למשל כשרוצים להציג את השיר בתצוגה הגדולה שלו).

compare_tsamps: סוג של כפילות קוד אבל לא באמת מכיוון שמתודה זו צריכה להיות מיושמת בצורה שונה כאן.

configurewin: מתודה שאחראית לשנות את החלון כאשר מגדילים או מקטינים אותו, ולהתאים הכל לגודל החדש, צריך לשפר אותה כי כאשר מקטינים את החלון יותר מידי דברים מתחילים קצת להשתגע, וכאשר הוא גדול מידי דברים נהפכים להיות קטנים מידי.

create_big_frame: מתודה שנקראת רק כאשר החלון מיוצר ובונה את הפריים הגדול בצורה בה הוא מוצג.

create_home: על אותה עקרון של המתודה הקודמת, מייצרת את ה"בית", החלון במצבו ההתחלתי, לפני שעושים כלום.

delete_playlist: מתודה שאחראית על מחיקת פלייליסטים במקרה שלוחצים עליהם מהרשימה במקש אמצעי של העכבר. במקרה שהפלייליסט אחרון זה די קל ואם הוא לא אחרון צריך להשלים את המקום שלו, אז זה מעלה את כל הפלייליסטים שמתחתיו(נמצאים אחריו ברשימה).

disable_textbox: נועד לבטל את החיפוש כאשר מבצעים לחיצה בכל מקום אחר, בשביל שיהיה אפשר להשתמש במקש הרווח לעצור ולהמשיך את השיר.

display_plsts: נועד להציג את הפלייליסטים בתוכנה לפי שמוצגים בעצם היוזר שקיים לממשק הגרפי הספציפי הזה.

enable_textbox: על אותו עקרון של הdisable רק שמפעיל כאשר לוחצים עליו.

find_dur: נועד למצוא את האורך של שיר מעצם של שיר.

load_first: נועד לטעון את עצם השיר הראשוני כאשר מתחילים את התוכנה (מעצם המשתמש שמצורף לתוכנה).

make_big_song_widget: מייצרת את העצם גדול של שיר, והמתודה הזאת נועדה בשביל להכין אותו במקרה שיש צורך בכך.(קיים לכל שיר, אך נראה רק במקרה שמפעילים את מתודת הוויואו (כאשר נלחץ על התמונה העגולה)).

make_circle_image: מייצר את התמונה העגולה שיש בצד ימין למטה של המסך.

make_crsobject: מייצר עצם של שיר מנוסח הJSON שמתקבל במשתמש שמגיע מהDB.

make_playlist: מראה את הפלייליסט שלוחצים עליו (קורה כאשר לוחצים על לייבל שמכיל פלייליסט).

make_playlist_name: מייצרת פלייליסט חדש מתוך שם שניתן לאפליקציה לאחר לחיצה על לייבל ה"פלייליסטס".

dson_enter, on_leave metho: מתודות שונעדו לשלוט על הצבעים של הלוגואים של הדילוג, דילוג לאחור, והפעלת השיר במידה והעכבר מעליהם הם הופכים לאפורים, ואם הוא איננו הם חוזרים לשחורים.

on_press methods: מתודות שמטרתן להעביר את השיר קדימה או אחורה או לנגן שיר או לעצור אותו.

play_next_song: די זהה למתודה שלה מסוג הon_press, ההבדל היחיד הינו שהמתודה הזאת נקראת כאשר שיר נגמר והשיר הבא מתחיל לעומת השניה שנקראת מתוך לחיצה על כפתור.

search_songs: מתודה שמטרתה לנקות את הקנבס של השירים ולחפש שירים חדשים בכל מאגרי המידע הקיימים.

seeking_action: מתודה שמטרתה להפוך משתנה אחד לחיובי כאשר מזיזים את הסליידר, על מנת שהתוכנה שתדע שהיא נמצא במצב של seeking.

shuffle_plsts: מתודה שמטרתה לערבל פלייליסט כלשהו(נקראת לאחר לחיצה על כפתור).

slidr_seek, slidr_volume_seek: מתודות שמטרתן לשלוט על הווליום לפי פס ההתקדמות ועל המיקום בשיר לפי הפס של השיר.

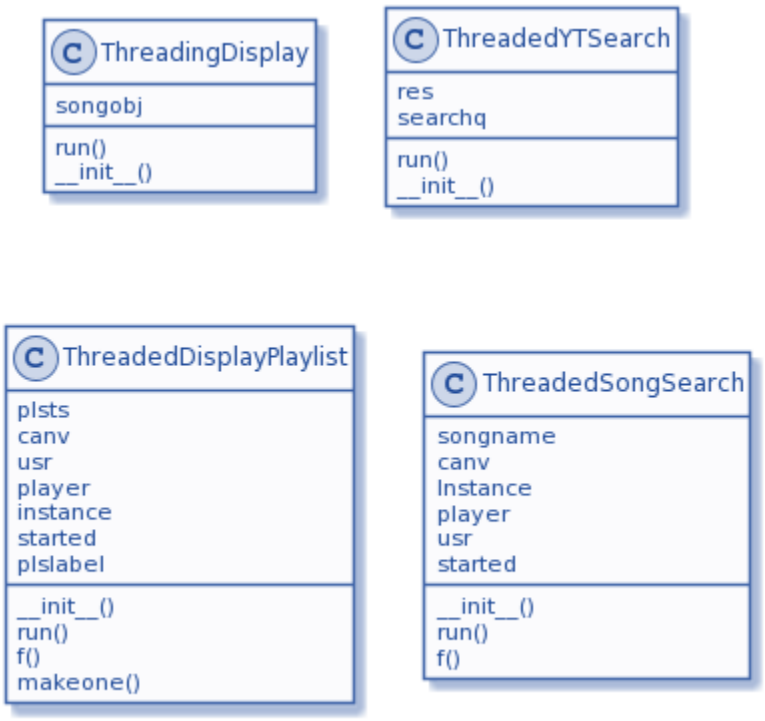
update_user: מתודה שמחלקתה לעדכן את עצם המשתמש לאחר התנתקות מהממשק.

view_big_song: מתודה שאחראית על הצגה של השיר הגדול במקרה ונקרא לעשות זאת.

view_search: מתודה שמטרתה להחליף למצב חיפוש השירים במקרה ורוצים לחזור חזרה.

הסבר אודות threadedclasses

הקובץ threadedclasses מכיל מגוון של מחלקות אשר מטרתן היחידה היא להחיל מנגון של threading על מחלקות אשר כבר קיימות בפרויקט, או פונקציות אשר רצות ונדרש שירוצו במקביל, לדוגמה, אם התוכנה רוצה לבצע חיפוש במאגר של ספוטיפיי על שיר, פעולה שלוקחת כמה שניות, באופן טורי על רשימת שירים דבר זה יקח המון זמן, אך לעומת זאת אם נריץ חיפוש במקביל בת'רדים שונים על כל האובייקטים ברשימה נמצא תוצאות לכולם תוך שניות בודדות.



ThreadedYTSearch

מחלקה זו אחראית על ה"ת'רדיזציה" של החיפוש ביוטיוב, המחלקה מקבלת בתור פרמטרים את:

res.1: מספר התוצאות שרוצים לקבל כאשר מבצעים חיפוש.
searchq.2: (שאילתת חיפוש, סרץ' קווירי) זהו סטרינג שמייצג את השאילה שיש לחפש ביוטיוב.

run.3: פונקציה שמריצה את שאילתת החיפוש ומחזירה את כל תוצאות החיפוש (באמצעות yield מכיוון שמקבלים מערך של תוצאות).

```
def run(self):
    """This function runs the thread"""
    y = YoutubeSearch(self.searchq, max_results=self.res)
    if self.res > 0:
        with concurrent.futures.ThreadPoolExecutor(max_workers=self.res) as executor:
            create_song_result = {executor.submit(one_search, y.to_dict()[x]['link']): x for x in range(0,
len(y.to_dict()))}
            for future in concurrent.futures.as_completed(create_song_result):
                try:
                    if future.result() is not None:
                        yield future.result()
                except Exception as exc:
                    exc_type, exc_obj, exc_tb = sys.exc_info()
                    print(">ERROR message:in making youtube song, ", exc_type, exc_obj, exc_tb, exc)
```

ThreadingDisplay

מחלקה זו אחראית על ה"ת'רדיזציה" של הצגת השירים, המחלקה מקבלת בתור פרמטרים את:

songobj.1: האובייקט שמייצג את השיר שיש להציג על המסך. מקבל GuiObj.

run.2: פונקציה שמריצה את משימת הצגת השיר בת'רד חדש.

```
def run(self):
    """This function runs the thread"""
    try:
        t = Thread(target = lambda : self.songobj.display())
        t.start()
    except Exception as e_rr:
        print(">ERROR message:in threaded_song_disp ",e_rr)
```

ThreadedSongSearch

מחלקה זו אחראית על ה"ת'רידזציה" של תהליך חיפוש השירים, מכניסה אותו לת'רד שונה בשביל שלא תתקע את הפעילות של הממשק הגרפי. המחלקה מקבלת שלל פרמטרים מכיוון שאחראית גם על הצגת השירים על פני הממשק ולכן צריכה לקבל מצביעים בזכרון על דברים שיש בממשק כמו הקנבס שעליו עליה להציג את השירים ועוד.

songname.1: שם השיר שעליו יש להריץ חיפוש.
canv.2: אובייקט הקנבס שעליו יש להציג את השירים.
Instance, player.3: אובייקטים אשר מקושרים לנגן הVLC שקיים בו שימוש בפרויקט.
usr.4: המשתמש שמבקש לבצע את החיפוש
started.5: פרמטר שצריך לבדוק אם המשתמש הזה כבר החל לנגן שיר כלשהו מאז שפתח את התוכנה.

run.6: מתודה שאחראית על הרצת החיפוש במערכת (הדאטהבייס) וביוטיוב גם כן ועל הצגת התוצאות.

```
class ThreadedSongSearch():
  """This class is made for threading the song searching task."""
  def __init__(self, songname, canv, usr, player, instance, started):
    self.songname = songname
    self.canv = canv
    self.Instance = instance
    self.player = player
    self.usr = usr
    self.started = started

  def run(self):
    def f():
      self.canv.component('canvas').delete('all')
      results = {
        'dbresult': []
      }
      try:
        if requests.get('http://idanbb.pythonanywhere.com').status_code == 200:  #checks if the online song
collection works
          """Searching in the database, took it out for being too long"""
          self.canv.image = []
          """Displaying the results"""
          for obj, i in zip(results['dbresult'][0:10], range(10, 10 + 160 * min(len(results['dbre-
sult']),self.usr.prefer_res), 160)):
            """This makes a thread for each results and displays it using the
            'ThreadedDisplay' method"""
      except Exception as e:
        print(">ERROR Messaege: Error in searching songs thread")
      ytsearch = ThreadedYTSearch(self.songname, self.usr.prefer_res - len(results['dbresult'])).run()
      x = list(ytsearch)
      ytobjs = []
      """Displays yourube results"""
    t = Thread(target=f)  # puts it all on a thread
    t.start()
```

ThreadedDisplayPlaylist

מחלקה זאת אחראית אודות הצגת הפלייליסט בת'רד נפרד כאשר רוצים להציג פלייליסט על המסך. מחלקה זו מקבלת בתור פרמטרים את הפלייליסט, את הקנבס שעליו יש להציג, את המשתמש שמבקש להציג את הפלייליסט, נתונים אודות הנגן, האם המשתמש כבר התחיל לנגן שיר לפני ואת הלייבל שעליו יש לכתוב את שם הפלייליסט.

plsts.1: הפלייליסט שיש להציג
canv.2: הקנבס שעליו יש להציגו
usr.3: המשתמש שמבקש להציג את הפלייליסט.
player, instance.4: נתונים אודות נגן הVLC.
started.5: בוליאני של האם המשתמש כבר ניגן שיר לפני כן?
plslabel.6: הלייבל (טקסט בממשק הגרפי) שאותו יש לשנות לשם הפלייליסט.

run.7: מתודה שאחראית על הצגת הפלייליסט ונכנסת לת'רד.

```
def run(self):
    try:
        self.canv.place(y=100, x=15)  # tries to place the canvas it case it wasnt placed
    except:
        pass

    def f():
        self.canv.component('canvas').delete('all')  # clears the canvas
        self.plslabel.configure(text=self.plsts.name.upper())  # configures the label
        self.plslabel.place(x=617, y=45, anchor=CENTER)  # places the label
        songs = [song for song in self.plsts.songs]  # creates a list of the songs from the playlist
        self.canv.image = []
        try:
            for obj, i in zip(songs,
                              range(10, 10 + 160 * len(songs), 160)):
                t = Thread(target = lambda x = obj, i=i : self.makeone(x,i))  # makes a thread for each song
                t.start()
        except Exception as e:
            exc_type, exc_obj, exc_tb = sys.exc_info()
            print(">ERROR message in ThreadedDisplayPlaylist: ",exc_type, exc_tb.tb_lineno, e)
    t = Thread(target=f)
    t.start()
```

makeone.8: מתודה שיודעת להציג שיר אחד ונעזרים בה במתודת הrun שתפקידה להציג פלייליסט שלם.

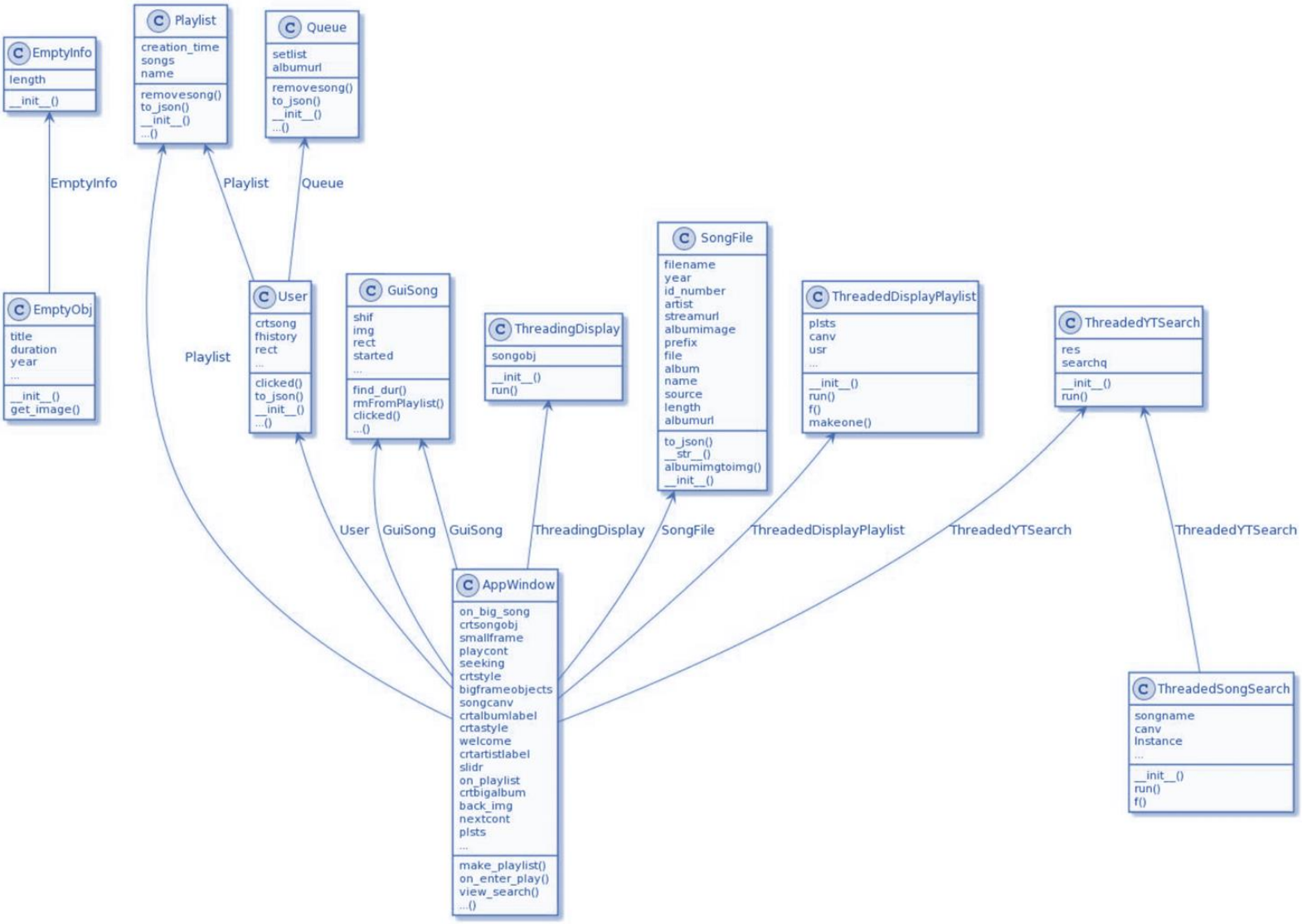
```
def makeone(self,obj,i):
    obj = from_crs_to_song(obj)
    obj.year = obj.album
    try:
        print(">>INFO: Displaying the songobj : ", str(obj.name))
    except:
        print(">ERROR message: Couldn't print the obj's name")
        pass
    guiobj1 = GuiSong(self.canv, obj, 10, 10 + i, self.player, self.instance, self.usr, self.started,
                      playst=self.plsts)
    ThreadingDisplay(guiobj1).run()
```

פונקצית from_crs_to_song

הפונקציה הנ"ל אחראית על הפיכת עצם json לעצם songfile, שמה ניתן לה מכיוון שלמשתנה הjson קוראים crs בתוכנה ומבצעת המרה.

```
def from_crs_to_song(crsong):
    """This fucntion turns a 'crsong' formatted json object into a songfile"""
    if 'yid' in crsong:  # checks if the song is in youtube format
        return SongFile(file=crsong['yid'],name=crsong['name'], artist=crsong['artist'], album=crsong['al-
bum'],albumurl=crsong['albumurl'], source='yt', prefix='mp3', streamurl=crsong['streamurl'])
    else:
        db_json_song = db.songdb.find_one({"_id": objectid.ObjectId(crsong['_id'])})  # if it isn't it just pulls it out
the db
        if db_json_song is None:
            print(">>INFO: The id of ",crsong['_id'], ' yielded None')
        return songfilefromDB(db_json_song)
```

תרשים UML יותר מורחב של כל העצמים בפרויקט

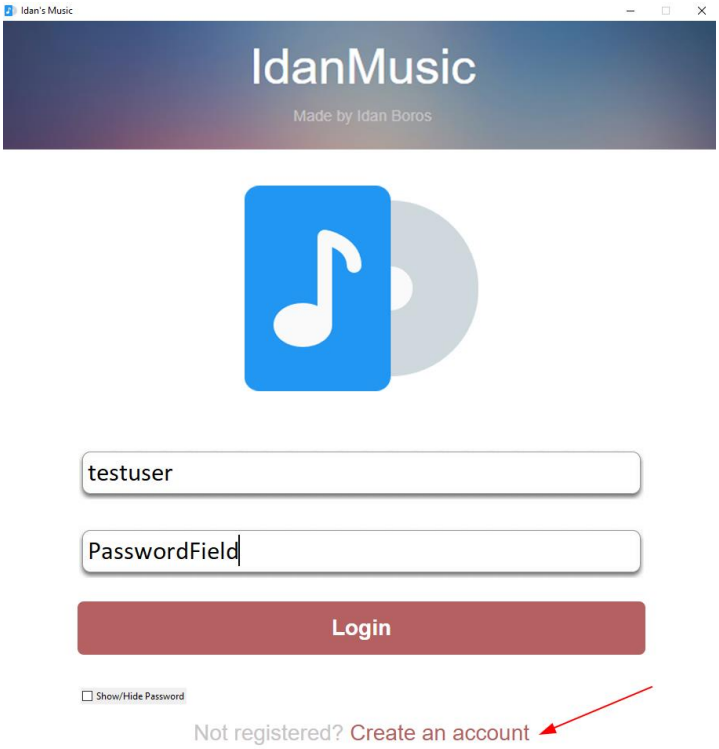


מדריך למשתמש – נלקח מתוך readme.md

Signing up and getting set up:

Signing Up!

1. Start the login.py file in the folder with the venv in the venv folder. It can be done through CMD using the command '/path/to/project/venv/Scripts/python.exe /path/to/project/login.py'
2. Wait a little, it can take a minute for it to form a connection with the cloud database.
3. Create your user! Enter your credentials and press the button here

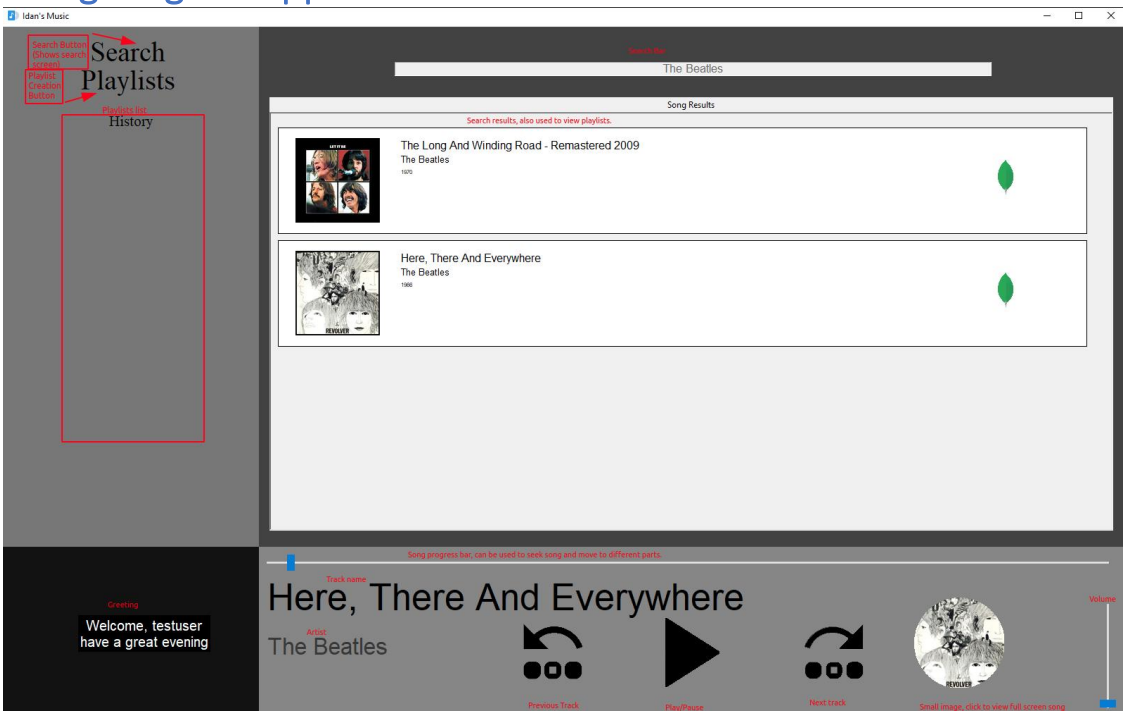


4. Fill how many songs you'd like to be viewed per search(Your prefrence)
5. Create the user.
6. Finished the signing up process!

Logging in!

1. Start the .exe file in the folder and wait a little.
2. fill in your credentials and make sure they are correct.
3. Press either enter or the login button.
4. Voila!

Navigating the app!



Searching AND Playing

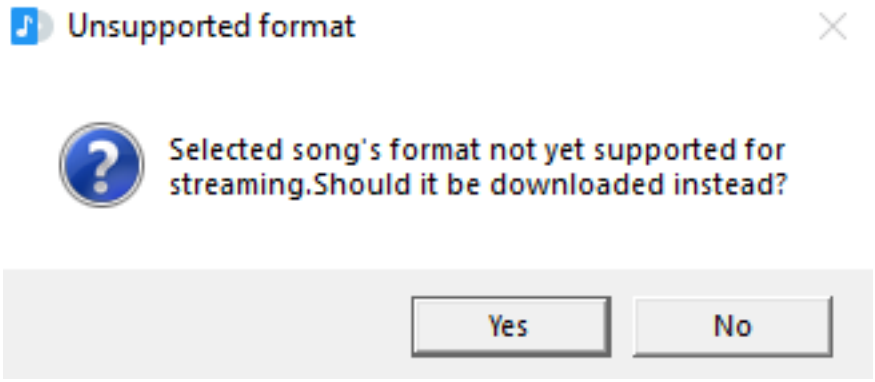
1. Use the search bar provided in the upper right section to perform a search.
2. Press the enter key to find results of search query specified in the bar.
3. Playing a song is possible by pressing the song with the left mouse button. Pausing/Playing is also possible by pressing the space button assuming the search box is not selected.
4. If you try to play a song while playing a different song, it'll add the second song into the user's queue and it'll play next.
5. Change volume using the volume slider in the bottom right corner.

Playlists

1. Create a new playlist by clicking the "Playlists" text box.and following further instructions inside the app.
2. Insert song into a playlist by right pressing it's object on screen, and then selecting a playlist.
3. Remove a song from a playlist by right pressing it from within a playlist.
4. Remove a playlist by pressing the middle mouse button on it(Right click is used for adding all the playlists song into the user queue)

Notes:

- Seeking is possible using the progress bar.
- Press the small circle image of the song to display the song in the “fullscreen” mode. Exit it by pressing either “Search” or entering a playlist.
- If you try to play an unplayable song a prompt will pop up asking if you’d like to save the song to your computer(Can be played using an external player), follow the instructions after the prompt - Example image added.



הסבר בסיס הנתונים

מאגר הנתונים נכתב בmongodb, הסתכלתי עליו דרך התוכנה שלהם אשר נקראת compass, אני לא ממש יודע איך אמורים לייצג שם טבלאות אז פשוט אצרף תרשימי UML של העצמים שמאוחסנים שם וגם תמונות ממאגר הנתונים העכשווי, קיימות 2 "טבלאות" (אני מניח שטבלאות זה המקביל בSQL) בפרויקט שלי, אחת של שירים והשניה של משתמשים, כאשר מעלים שיר למאגר (בpythonanywhere) נשלחת שאילתה לבסיס הנתונים אשר מוסיפה את השיר למאגר השירים, ל"טבלת" השירים, השירים מתאחסנים בתור עצמי JSON והנה דוגמה מהמאגר לשיר עם הסבר אודות כל "עמודה".

songDB


```
_id: ObjectId("5eabe468ed32b5e6d0b54fe9")
file: "./song_library/the_long_and_winding_road_remastered_2009.mp3"
name: "The Long And Winding Road - Remastered 2009"
artist: "The Beatles"
album: "Let It Be (Remastered)"
year: "1970"
albumimage: Binary('/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRofHh0aHBwgJC4nICIsIxwcKDcpLDAx...', 0)
length: "218.4179419868809"
prefix: ".mp3"
streamurl: "http://idanbb.pythonanywhere.com/uploads/the_long_and_winding_road_-_r..."
```


_id.0: המזהה של השיר הזה ספציפית במאגר.

file.1: המיקום של השיר באופן יחסי בתוך השרת.
name, artist, album, year, length, albumimage.2: פרטים אודות השיר, נועדו על מנת להציגו.
prefix.3: הסיומת של קובץ השיר, נועדה על מנת לדעת אם ניתן להסטרים אותו.
streamurl.4: הקישורית לסטרימינג של השיר במאגר, נועדה על מנת לנגן שירים.

usrDB

"טבלה" שנועדה לייצג את המשתמשים בשירות, כאשר מייצרים משתמש נשלח המשתמש לטבלה זו ומתווסף עצם json שלו וכאשר מנסים להתחבר המשתמש נשלף משם וככה מבצעים בדיקה של האם המשתמש נכון, המשתמש גם מאחסן מידע רב כמו היסטוריית שמיעה, פלייליסטים ועוד.

הסבר אודות כל עמודה:

```
_id: ObjectId("5ee73e8c0ba677a69949adbc")
username: "kobi_shutzman"
password: "c195aa55f604c8a8286f6dba9badbfd5dac692983bcdd8796f77098896dd0e9cac6e44..."
prefer_res: 30
› playlists: Array
› queue: Object
› history: Array
› fhistory: Array
crtsong: null
date: "2020-06-15T12:24:54.274665"
```

_id.0: המזהה של המשתמש הזה ספציפית במאגר.

username.1: שם המשתמש.
password.2: סיסמת המשתמש, לאחר גיבוב SHA
prefer_res.3: כמה שירים המשתמש רוצה שיופיעו כאשר מחפש שיר.
playlists.4: מערך שמכיל את כל הפלייליסטים של המשתמש, מערך של אובייקטי JSON שמייצגים פלייליסטים.
queue.5: תור השירים של המשתמש, איזה שירים אמורים להתנגן להבא?
history.6: היסטוריית ההשמעה של המשתמש, למקרה שירצה לחזור אחורה.
fhistory.7: ההיסטורייה המלאה, כי אם חוזרים בהיסטוריה אחד אז הוא יוצא מההיסטוריה(לנגן שיר קודם), לעומת זאת מכאן הוא איננו יוצא.
crtsong.8: השיר אשר מתנגן ברגע זה אצל המשתמש.
date.9: תאריך היצירה של המשתמש.

רפלקציה

וואו, מאיפה להתחיל?

העבודה על הפרויקט הזה היה הדבר שהעסיק אותי הכי הרבה מבחינת לימודים עוד מאז שהתחלתי לעבוד עליו, בערך לפני תשעה חודשים, העבודה הייתה כל כך מעניינת וגרמה לי להשאר ער לילות רבים עד שעות מאוחרות רק בניסיון לפתור בעיות ותקלות, היא גרמה לי להבין שאם אני רוצה משהו הוא אפשרי, ושלכל דבר יש פתרון, עם כמה שזה נשמע מוגזם.

בפרויקט קיבלתי המון המון ידע שרכשתי בעצמי דרך האינטרנט ודרך חקירה רבה על הנושאים שעניינו אותי, שמעתי המון מילים בתחום המחשבים שלא הבנתי ואני יכול להגיד באופן חד משמעי שעכשיו אחרי שסיימתי את הפרויקט (לא באמת סיימתי, אני עומד להמשיך לעבוד עליו גם לאחר ההגשה) שהידע שלי הרבה יותר נרחב מאשר כשהתחלתי. עם זאת, אני מודע לכך שיש עוד המון המון מה ללמוד.
במהלך העבודה על הפרויקט נתקלתי בהמון קשיים וסיבוכים, ראשית, רציתי לעבוד עם טכנולוגיות שלא היה לי ידע עליהן כלל, והרעיונות שהיו לא לא נראו כל כך ישימים עם חומר שהיה בכיתה.
ראשית, הפרויקט הראשוני שלי היה שונה לגמרי ועבדתי עליו בערך חצי שנה, אבל לאחר שסיימתי איתו החלטתי שאני לא מסופק ממנו, הפרויקט היה בוט לשרת דיסקורד שיכול ליישם כל מיני דברים אבל זה לא רלוונטי אז לא אפרט אודותיו, אך אציין כי גם במהלך עבודתי עליו חקרתי את כל הנושא לא במסגרת בית הספר רק מתוך עניין בו, וכך גם היה בפרויקט שאני מגיש היום, בפרויקט נעשה שימוש במבנה נתונים מסוג שלא למדנו עליו בבית הספר (NoSQL), נעשה יישום לרשתות בדרך שלא נלמדה בבית הספר (למדנו אודות עבודה בלעדית מול sockets, ובפרויקט שלי נושא התקשורת נעשה על ידי שימוש בflask וrequests, פרוטוקולים השונים מתכנות sockets).
במהלך הפרויקט אני חושב שהאתגר הראשון שנתקלתי בו היה כזה "אני רוצה שאוכל להעלות שירים ואז לנגן אותם, אז איך אוכל לבצע העלאה לשרת?", פתרון זה נפתר לאחר מחקר על ידי שימוש בשרת rest שכתבתי באמצעות המודול flask.
האתגר השני היה "איך אנגן את השירים ברגע שהועלו לשרת?" פתרון זה נפתר על ידי שימוש בביינדים של VLC בפייתון.
עוד אתגרים קרו כאשר עבדתי על תכנון הGUI, מכיוון שלא למדנו על הכנת GUI ברמת סיבוך כזאת בבית הספר, אך פתרתי אותו באמצעות המון שימוש ברשת, כמו כן לאחר שסיימתי עם מנגנון ההעלאה וההשמעה הגעתי למסקנה שזה לא מספיק לי, ובגלל זה גם הוספתי את האימפלמנטציה של השמעת שירי יוטיוב דרך התוכנה, דבר זה הביא עימו אתגר, איך אדע פרטים על השיר חוץ מהכותרת שלו דרך יוטיוב? הפתרון לאתגר זה היה שימוע בAPI של ספוטיפיי על מנת לדלות מידע אודות השיר.

המסקנה שלי בנוגע לכל התהליך היא שכתיבת הפרויקט הזה פיתחה את צורת המחשבה שלי, מכיוון שהיום אני חושב בצורה יותר הגיונית ולוגית מאשר לפני הכתיבה שלו, כמו שארחיב בפסקה הבאה, וגם הקנתה לי המון כלים שגיליתי בצורה עצמאית.

אילו הייתי מתחיל לעבוד על הפרויקט היום הייתי משנה את הגישה שהייתה לי למאגר הנתונים, באותו הזמן חשבתי שאני עובד בצורה מאוד יעילה אבל היום אני מבין שלאנדקס כל שיר יוטיוב שנשמע בתור שיר שנמצא במאגר יהיה הרבה יותר יעיל מאשר לשמור כל שיר בתור עצם בהיסטוריה, דבר מאוד בזבזני, הייתי הולך על דרך הרבה יותר חסכונית, אולי אעשה זאת בעתיד אבל היום מאגר הנתונים שלי די בזבזני, דבר שהייתי משנה מלכתחילה לו הייתי יכול, בין היתר הייתי גם כותב את כל הקוד בצורה יותר מאורגנת מלכתחילה, מכיוון שבכתיבת הספר שיניתי המון קוד שהיה די בלתי קריא ואם מלכתחילה היה קריא זה היה מאוד עדיף.

אם הייתי עושה זאת אז הפרויקט היה עובד יותר מהר, וגם הייתי בונה מאגר מידע משלי אודות שירים עם פרטים עליהם, אמנם לא היה יעיל ב100% בגלל תוצאות שגויות מהAPI של ספוטיפיי אבל הפרויקט היה מצליח לשלוף מידע מהר יותר מכיוון שהDB של מונגו מהיר יותר מחיפוש ביוטיוב ב99% מהמקרים.

נספחים

הסבר אודות דבר אחד שלא ממש יצא לי להסביר עליו במהלך כתיבת הספר, Pmw, הכלי הזה הינו מודול לפייתון שלא ממש מוכר ומכיל בתוכו "מגה-ווידג'טס", מטרתו היא מעין הרחבה של המודול הקיים Tkinter מכיוון שמכיל בתוכו ווידג'טים שמורכבים ממספר ווידג'טים של tkinter, מעין "מכליות" שמכילות מספר ווידג'טים, במהלך הפרויקט שלי נעזרתי במגה ווידג'ט אחד בשם ScrolledCanvas, והוא הקנבס שעליו מופיעות תוצאות השירים, נתקלתי במודול זה במקרה לחלוטין כאשר סתם עיינתי בספר Tkinter שקובי העלה לאתר שלו בשם " Python and Tkinter Programming", דרכו למדתי אודות מודול זה וכך בעצם הגעתי ללהשתמש בו אף על פי שהוא לא מוכר ממש ואין לו הרבה תמיכה באינטרנט.

פיצ'רים עתידיים שאשמח להוסיף

האופציה להציג יוזר מיוזר אחר (קיימת כבר, אך ארצה לשפר ולהוסיף אופציה לראות את הפלייליסטים של אותו יוזר)

עצמים של אלבומים שיכילו שירים, הצגתם ושיוכם לעצמי "אמנים" שיכילו תחתם אלבומים. היה לי את הרעיון הזה עוד מלכתחילה אך לא יישמתי אותו מכיוון שנראה לי מורכב מידי במסגרת הזמן שהייתה לי.

שיפור של הדאטהבייס שיאנדקס כל שיר יוטיוב שכל אדם באפליקציה יבחר לשמוע על מנת שבעתיד שאדם אחר ירצה לשמוע יקבל תוצאה מהר יותר, ככה אבנה דאטה בייס של שירים משל עצמי מבלי הצורך להעלות כל שיר.

ביבליוגרפיה

במהלך הכתיבה השתמשתי בעיקר בידע שרכשתי דרך האינטרנט במדריכים, אני לא ממש שמרתי את הקישורים של כולם ואני לא בטוח בנוגע איך לכתוב זאת בפורמט APA, אני מניח שהחלק הזה פחות רלוונטי בנוגע לפרויקט שלי אבל אני כן יכול לצרף לכאן את הספר שהשתמשתי בו וכתבתי אודותיו בחלק ה"נספחים".

1. Grayson, J. (2000). *Python and tkinter programming*. Shelter Island, New York:Manning.