```
import numpy as np
import csv, string
from sklearn.model_selection import train_test_split
from sklearn import preprocessing, neighbors
from scipy import spatial
from Users import User
import random

class Song:
    def __init__(self, songid, title, release, artist_name, year):
        self.songid = songid
        self.title = title
        self.release = release
        self.artist_name = artist_name
        self.year = year

class OtherUser:
     def __init__(self, userid, songid, rate):
        self.userid = userid
        self.songid = songid
        self.rate = rate

def loadUsers():
    users = []
    file = open("music_dataset/10000.txt")
    for line in file.readlines():
        pieces = line.split()
        if (len(pieces) == 3):
            users.append(OtherUser(pieces[0].strip('" \t'), pieces[1].strip('" \t'), pieces[2].strip('"
\t')))
    return users

def loadSongs():
    print("loading data...")
    songs = []
    file = open("music_dataset/song_data.csv")
    for line in file.readlines():
        pieces = line.split(",")
        if (len(pieces) == 5):
            songs.append(Song(pieces[0].strip('" \t'), pieces[1].strip('" \t'), pieces[2].strip('" \t'),
pieces[3].strip('" \t'), pieces[4].strip('" \t')))
    return songs

def songsContainsArtist(songs, artist):
    for song in songs:
        if song.artist_name == artist:
            return True
    return False

def printArtistSongs(songs, artist):
    print("\n======All Songs by ",artist,"======")
    for song in songs:
        if song.artist_name == artist:
            print(song.title)

import operator
def getSimilarListener(songid):
    print("loading")
    f = open("music_dataset/10000.txt", encoding="utf8")
    dic = {}
    count = 0
    for i in f.readlines():
        if(count >= 5000):
            break
        else:
            count+=1
        l = i.strip("\n").split("\t")
        if(l[1].strip()==str(songid)):
            if(dic.get(l[0].strip()) == None):
                dic[l[0].strip()] = [l[2]]
    sorted(dic.items(), key=lambda x: x[1], reverse=True)
    len_list=len(list(dic.keys()))
    if(len_list != 0):
        return list(dic.keys())[0]
    print("No similar listener exists!")
    return None


def displaySongsByListener(users, songs, listenerID):
    anst = input("Do you want the song list shared by other listeners(Similar Preference)? (y/n):
")
    if(anst == "n"):
        return
    print("\n======Song List By Listener======")
    print("{:<14}{:<50}".format('Listener ID','Song Title'))
    print("{:<14}{:<50}".format(listenerID, ''))
    for user_i in users:
        if user_i.userid == listenerID:
            song_id=user_i.songid
            for s in songs:
                if s.songid == song_id:
                    print("{:<14}{:<50}".format('',s.title))

def randomSong(songs):
    mx = len(songs)
    index = random.randrange(1, mx)
    print(songs[index].title, " by ", songs[index].artist_name)
    answ = input("Do you like it? (y/n):  ")
    if(answ == "y"):
        anst = input("Do you want all songs by this artist? (y/n):  ")
        if(anst == "y"):
            printArtistSongs(songs, songs[index].artist_name)

#list of users
users = []

#read a single line of data from users.csv
def processLine(line):
    try:
        l = line.split(",")
        username = l[0]
        password = l[1]
        playlist = l[2].split("|")
        songs = []
        for song in playlist:
            if len(song) > 0:
                songs.append(song)
        user = User(username, password, songs)
        return user
    except:
        return False
#read in user data from users.csv
def loadUserData():
    try:
        f = open("users.csv", "r")
        for i in f.readlines():
            user = processLine(i.strip())
            if user != False:
                users.append(user)
        print("Data sucessfully loaded!")
    except:
        print("users.csv does not exist yet")
```

```
#write user data to users.csv
def writeUserData():
    f = open("users.csv", "w")
    for user in users:
        f.write(str(user) + "\n")
    print("Data succesfully written!")

#does a user with this username exist?
def userExists(username):
    for user in users:
        if(user.getUsername() == username.lower()):
            return user
    return False

#either return the user associated with this username, or False
def validateUsername(username):
    user = userExists(username)
    if(user != False):
        password = input("Please enter the users password: ")
        if(user.validatePassword(password)):
            return user
        else:
            print("Invalid password")
            return False
    return False

#make the dictionary of SONGID->SONG TITLE
def makeUsersDic():
    print("loading user data...")
    f = open("music_dataset/song_data.csv", "r")
    dic = {}
    for i in f.readlines():
        l = i.split(",")
        dic[l[0].strip()] = l[1]
    print("user data loaded...")
    return dic

#create a new user
def createNewUser():
    print("CREATE NEW USER")
    username = input("Enter a username: ")
    exists = userExists(username)
    if(exists == False):
        password = input("Enter a password: ")
        user = User(username, password)
        users.append(user)
        print("User succesfully added!")
    else:
        print("That user already exists")

#make two dictionaries, song->list of users, user->list of songs
def makeSongDic():
    print("loading song data...")
    f = open("music_dataset/10000.txt", "r")
    dic = {}
    usersdic = {}
    count = 0
    for i in f.readlines():
        if(count >= 5000):
            break
        else:
            count+=1
        l = i.strip("\n").split("\t")
        if(dic.get(l[1].strip()) != None):
            dic[l[1].strip()].append(l[0])
        else:
            dic[l[1].strip()] = [l[0]]
        if(usersdic.get(l[0]) != None):
            usersdic[l[0]].append(l[1].strip())
        else:
            usersdic[l[0]] = [l[1]]
    print("song data loaded...")
    return (dic, usersdic)

#ceneted_cosine of two lists
def centered_cosine(l1, l2):
    usersList = []
    for user in l1:
        if(not (user in usersList)):
            usersList.append(user)
    for user in l2:
        if(not (user in usersList)):
            usersList.append(user)
    numList1 = []
    numList2 = []
    for user in usersList:
        numList1.append(l1.count(user))
        numList2.append(l2.count(user))
    return 1 - spatial.distance.cosine(numList1, numList2)

#find n most similar songs to a given song
def mostSimilar(songs, songID, n):
    testData = songs[songID]
    ratings = []
    count = 0
    for song in songs:
        if(count < 10):
            data = songs[song]
            cc = centered_cosine(testData, data)
            ratings.append((song, cc))
            count+=1
        else:
            break
    def func(e):
        return e[1]
    ratings.sort(key=func)
    ratings.reverse()
    return ratings[:n]

#display 100 new songs
def displaySongs(songs, dic, start):
    print("{:<14}{:<35}".format('Song ID','Song Title'))
    for i in range(start, start+100):
        songID = songs[i]
        print("{:<35} {:<14}".format(songID, dic[songID]))

#display a single new song
def displaySong(songs, dic, start):
    print("{:<14}{:<35}".format('Song ID','Song Title'))
    for i in range(start, start+1):
        songID = songs[i]
        print("{:<35} {:<14}".format(songID, dic[songID]))

#main function for displaying 100 new songs
def display(songs, dic):
    kg = True
    curr = 0
    while(kg):
        print("\n------------\n")
        displaySongs(songs, dic, curr)
        print("\n------------\n")
        ans = input("Would you like to see another 100 songs? (y/n):  ")
        if(ans == "n"):
            kg = False
            break
        else:
            curr+=100

#take a list of songs, displays them in a table
def displayResults(l, songs):
    print("\n*********Similar Songs********")
    print("{:<14}{:<35}".format('Song ID','Song Title'))
    for i in l:
        print("{:<35} {:<14}".format(i[0], str(songs[i[0]])))

#let a user add songs to his or her playlist
def make_playlist(user, songs, dic):
    print("\n")
    print("PLAYLIST EDITOR")
    kg = True
    curr = 0
    playlist = []
    while(kg):
        if(user.songInPlaylist(songs[curr])):
            curr+=1
            continue
        print("\n------------\n")
        displaySong(songs, dic, curr)
        print("\n------------\n")
        add = input("Would you like to add this song to your Playlist? (y/n):  ")
        if(add == "y"):
            user.addSong(songs[curr])
        ans = input("Would you like to see another song? (y/n):  ")
        if(ans == "n"):
            kg = False
            break
        else:
            curr+=1

#displays a users playlist for them
def viewPlaylist(dic, user):
    print("\n")
    print(user.getUsername() + "'s Playlist")
    print("{:<8}{:<20}{:<45}".format('Index','Song ID','Song Title'))
    playlist = user.getPlaylist()
    count = 0
    for song in playlist:
        print("{:<8}{:<20}{:<45}".format(str(count),song,dic[song]))
        count+=1

#returns the menu string
def getMenuStr():
    menustr = "\n-------------------\n"
    menustr += "What would you like to do?\n"
    menustr += "1.Find similar music.\n"
    menustr += "2.Create New User\n"
    menustr += "3.Login\n"
    menustr += "4.Get music suggestions for you\n"
    menustr += "5.Add to your Playlist\n"
    menustr += "6.View your playlist\n"
    menustr += "7.Lookup a song\n"
    menustr += "8.Display Songs List\n"
    menustr += "9.Get a random song\n"
    menustr += "10.Look up an Artist\n"
    menustr += "11.Quit\n"
    menustr += "---------------\n"
    return menustr

#finds to 10 users most similar to a given user, suggests songs they like
def knn(usersDic, user):
    test = user.getPlaylist()
    user_names = []
    classifier = []
    for name in usersDic:
        user_names.append(name)
        classifier.append(centered_cosine(test, usersDic[name]))
    userNamesArr = np.array(user_names)
    X_train = userNamesArr.reshape(len(user_names),1)
    y_train = np.array(classifier)
    lab_enc = preprocessing.LabelEncoder()
    thisUser = [user.getUsername(),user.getUsername()]
    thisUserArr = np.array(thisUser)
    X_test = thisUserArr.reshape((2,1))
    y_test = np.array([1.0,1.0])
    clf = neighbors.KNeighborsRegressor()
    X_train = lab_enc.fit_transform(userNamesArr.reshape(len(user_names),))
    X_train = X_train.reshape(len(user_names), 1)
    X_test = X_test.reshape(2,)
    X_test = lab_enc.fit_transform(X_test)
    X_test = X_test.reshape(2,1)
    y_train = y_train.reshape(len(classifier),)
    y_train = y_train.ravel()
    clf.fit(X_train,y_train.ravel())
    y_test = y_test.reshape(2,)
    accuracy = clf.score(X_test, y_test)
    predictFor = np.array([user.getUsername()])
    predictFor = lab_enc.fit_transform(predictFor)
    predictFor = predictFor.reshape(1,1)
    prediction = clf.predict(predictFor)
    nabes = clf.kneighbors(predictFor, 10)
    neighborsList = nabes[1].ravel().tolist()
    mostSimilarUsers = []
    for neighbor in neighborsList:
        name = user_names[neighbor]
        mostSimilarUsers.append(name)
    song_suggestions = []
    for i in mostSimilarUsers:
        if(len(song_suggestions) > 9):
            break
        else:
            songs = usersDic[i]
            for song in songs:
                if(not user.songInPlaylist(song)):
                    song_suggestions.append(song)
    return song_suggestions[:10]

#wrap up knn
def musicSuggestions(user, dic, songs):
    print("\n----MUSIC SUGGESTIONS FOR " + user.getUsername() + "----\n")
    if(len(user.getPlaylist()) == 0):
        print("User has added no songs yet...")
        return
    else:
        l = knn(dic, user)
        print("{:<8}{:<20}{:<45}".format('Index','Song ID','Song Title'))
        playlist = user.getPlaylist()
        count = 0
        for song in l:
            print("{:<8}{:<20}{:<45}".format(str(count),song,songs[song]))
            count+=1
        add = input("Would you like to add these songs to your playlist (y/n): ")
        if(add == "y"):
            for song in l:
```

```
            user.addSong(song)
        print("songs added!")

#main loop function
def main():
    songs = loadSongs()
    users = loadUsers()
    print("Welcome to the Music Recommender System!")
    currentUser = None
    userData = makeUsersDic()
    tup = makeSongDic()
    songData = tup[0]
    userPLHist = tup[1]
    songKeys = list(songData.keys())
    menu_str = getMenuStr()
    keep_going = True
    loadUserData()
    while(keep_going):
        if(currentUser != None):
            print("\nLogged in as: " + currentUser.getUsername())
        option = int(input(menu_str))
        if(option == 1):
            idSong = input("Please give the id of the song you would to find similar music to: ")
            numSongs = int(input("Please enter the number of similar songs you would like: "))
            if(songData.get(idSong) == None):
                print("Invalid id")
            else:
                similar = mostSimilar(songData, idSong, numSongs)
                displayResults(similar, userData)
        elif(option == 2):
            print("\n")
            createNewUser()
        elif(option == 3):
            print("\n")
            print("LOGIN")
            username = input("Please enter a username: ")
            chosenUser = validateUsername(username)
            if(chosenUser != False):
                currentUser = chosenUser
        elif(option == 4):
            if(currentUser != None):
                musicSuggestions(currentUser, userPLHist, userData)
            else:
                print("You are not logged in yet")
        elif(option == 5):
            if(currentUser != None):
                make_playlist(currentUser, songKeys, userData)
            else:
                print("You are not logged in yet")
        elif(option == 6):
            if(currentUser != None):
                viewPlaylist(userData, currentUser)
            else:
                print("You are not logged in yet")
        elif(option == 7):
            idSong = str(input("Please give the id of the song you want to look up: "))
            if(userData.get(idSong) == None):
                print("Invalid id")
            else:
                print(userData[idSong])
            similar_Listener=getSimilarListener(idSong)
            if(similar_Listener != None):
                displaySongsByListener(users,songs, similar_Listener)
        elif(option == 8):
            display(songKeys, userData)
        elif(option == 9):
            print("\n")
            randomSong(songs)
        elif(option == 10):
            print("\n")
            artistname = input("Please give the name of the artist you want to look up: ")
            artistname.replace("\r", "")
            artistname.replace("\n", "")
            artistname.strip('" \t')
            if(songsContainsArtist(songs, artistname)):
                printArtistSongs(songs, artistname)
            else:
                print("Invalid artist")
        elif(option == 11):
            print("goodbye!")
            keep_going = False
            break
        else:
            print("Invalid menu option")
    writeUserData()

main()
```