**Course Code: INT 404 Assignments: 3**

**Group: 6 Section: K18GE**

**Topic: A Music Recommendation System.**

**Team Members:**

**M Sai Prakash-11814552-53**

**Amal K.P-11811365-4**

**A Vishnu Teja Reddy-11811604-62**

**Ch Uday-11811606-52**

**Introduction:**

Recommendation is widely used in our daily life. Especially in the E-Commerce area, a good recommendation system can help users a lot. In today’s World, every customer is faced with multiple choices. For example, if I’m looking for a book to read without any specific idea of what I want, there’s a wide range of possibilities how my search might pan out. I might waste a lot time browsing around on the internet and trawling through various sites hoping to strike gold, I might look for recommendations from other people.

But if there was a site or app which could recommend me books based on what i have read previously, that would be a massive help. Instead of wasting time on various sites, i could just log in and select the books according to my taste.

Likewise, as we did selection of books in the above, we also do for music. There are many music lovers who will listen music to get there mind refreshed. In fact they also have taste in listening the songs. So if they select the specific songs they required, it make a take too much time, instead of that if we have any app or site to recommend to prefer to select our site or app.

**Algorithms used in this Recommendation System:**

**Popularity Model:**

It is the most basic and simple algorithm. We find popularity of each song by looking into the training set and calculating the number of users who had listened to this song. Songs are then sorted in the descending order of their popularity. For each user, we recommended top most popular songs except those already in his profile. This method involves no personalization and some songs may never be listened in future.

**Collaborative Model:**

Collaborative filtering involves collecting information from many users and then making predictions based on some similarity measures between items. This can be classified as user based and item based models. In item based model, it is assumed that songs that are often listened together by some users tend to be similar and are more likely to be listened together in future also by some other user. According to user based similarity model, users who have similar listening histories, i.e. have listened to the same songs in the past tend to have similar interests and will probably listen to the same songs in future too. So, by using history of user listening the songs, the recommendations will provided.

**Blocked Diagram:**

**Music Files**

**Extracting music properties**

**Database Extracting Compare using Make playlist of**

**Records Algorithm music on phone.**

**Code for project:**

# Importing Required Modules & libraries

from tkinter import \*

import pygame

import os

# Defining MusicPlayer Class

class MusicPlayer:

# Defining Constructor

def \_\_init\_\_(self,root):

self.root = root

# Title of the window

self.root.title("Music Player")

# Window Geometry

self.root.geometry("1000x200+200+200")

# Initiating Pygame

pygame.init()

# Initiating Pygame Mixer

pygame.mixer.init()

# Declaring track Variable

self.track = StringVar()

# Declaring Status Variable

self.status = StringVar()

# Creating Track Frame for Song label & status label

trackframe = LabelFrame(self.root,text="Song Track",font=("times new roman",15,"bold"),bg="grey",fg="white",bd=5,relief=GROOVE)

trackframe.place(x=0,y=0,width=600,height=100)

# Inserting Song Track Label

songtrack = Label(trackframe,textvariable=self.track,width=20,font=("times new roman",24,"bold"),bg="grey",fg="gold").grid(row=0,column=0,padx=10,pady=5)

# Inserting Status Label

trackstatus = Label(trackframe,textvariable=self.status,font=("times new roman",24,"bold"),bg="grey",fg="gold").grid(row=0,column=1,padx=10,pady=5)

# Creating Button Frame

buttonframe = LabelFrame(self.root,text="Control Panel",font=("times new roman",15,"bold"),bg="grey",fg="white",bd=5,relief=GROOVE)

buttonframe.place(x=0,y=100,width=600,height=100)

# Inserting Play Button

playbtn = Button(buttonframe,text="PLAY",command=self.playsong,width=6,height=1,font=("times new roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=0,padx=10,pady=5) # Inserting Pause Button

playbtn = Button(buttonframe,text="PAUSE",command=self.pausesong,width=8,height=1,font=("times new roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=1,padx=10,pady=5) # Inserting Unpause Button

playbtn=Button(buttonframe,text="UNPAUSE",command=self.unpausesong,width=10,height=1,font=("timesnewroman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=2,padx=10,pady=5)

# Inserting Stop Button

playbtn = Button(buttonframe,text="STOP",command=self.stopsong,width=6,height=1,font=("times new roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=3,padx=10,pady=5)

# Creating Playlist Frame

songsframe = LabelFrame(self.root,text="Song Playlist",font=("times new roman",15,"bold"),bg="grey",fg="white",bd=5,relief=GROOVE)

songsframe.place(x=600,y=0,width=400,height=200)

# Inserting scrollbar

scrol\_y = Scrollbar(songsframe,orient=VERTICAL)

# Inserting Playlist listbox

self.playlist = Listbox(songsframe,yscrollcommand=scrol\_y.set,selectbackground="gold",selectmode=SINGLE,font=("times new roman",12,"bold"),bg="silver",fg="navyblue",bd=5,relief=GROOVE)

# Applying Scrollbar to listbox

scrol\_y.pack(side=RIGHT,fill=Y)

scrol\_y.config(command=self.playlist.yview)

self.playlist.pack(fill=BOTH)

# Changing Directory for fetching Songs

os.chdir("/home/sameer/Desktop/CodeSpeedy/cs10/songs")

# Fetching Songs

songtracks = os.listdir()

# Inserting Songs into Playlist

for track in songtracks:

self.playlist.insert(END,track)

# Defining Play Song Function

def playsong(self):

# Displaying Selected Song title

self.track.set(self.playlist.get(ACTIVE))

# Displaying Status

self.status.set("-Playing")

# Loading Selected Song

pygame.mixer.music.load(self.playlist.get(ACTIVE))

# Playing Selected Song

pygame.mixer.music.play()

def stopsong(self):

# Displaying Status

self.status.set("-Stopped")

# Stopped Song

pygame.mixer.music.stop()

def pausesong(self):

# Displaying Status

self.status.set("-Paused")

# Paused Song

pygame.mixer.music.pause()

def unpausesong(self):

# Displaying Status

self.status.set("-Playing")

# Playing back Song

pygame.mixer.music.unpause()

# Creating TK Container

root = Tk()

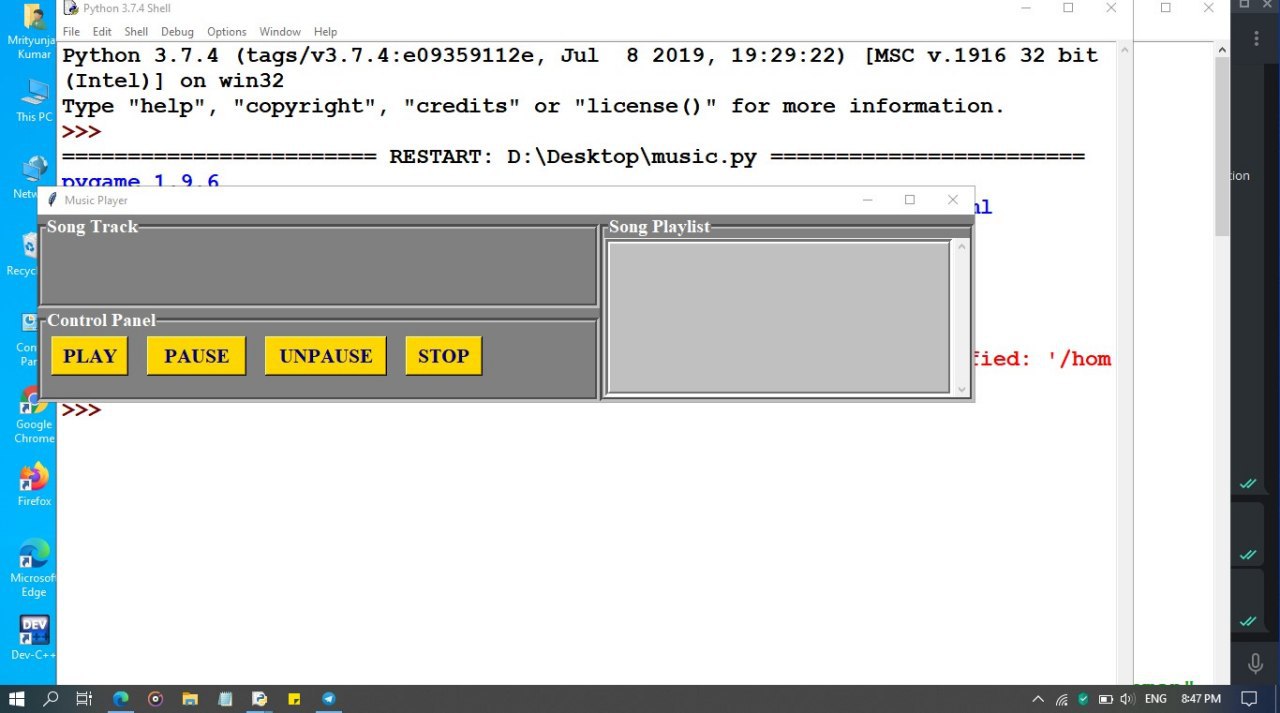
# Passing Root to MusicPlayer Class

MusicPlayer(root)

# Root Window Looping

root.mainloop()

**OUTPUT:**

****

**Conclusion:**

To summarize, we have learned about all the existing systems which included spotify, ganna.com and saavan. We found out ways in which these systems can influence our system. Spotify helped us understand how to use content based analysis in an effective way. Ganna.com helped us understand how to attract our target audience, and saavn helped us understand what kind of features we can add to our system. We decided to build a system to implement content based on his likes and dislikes. This data of the user will help our algorithm further to make the recommendations better.