Name:- Kamal Singh Rathore

Section: DE327

Roll no: 28

Regno: 12302604

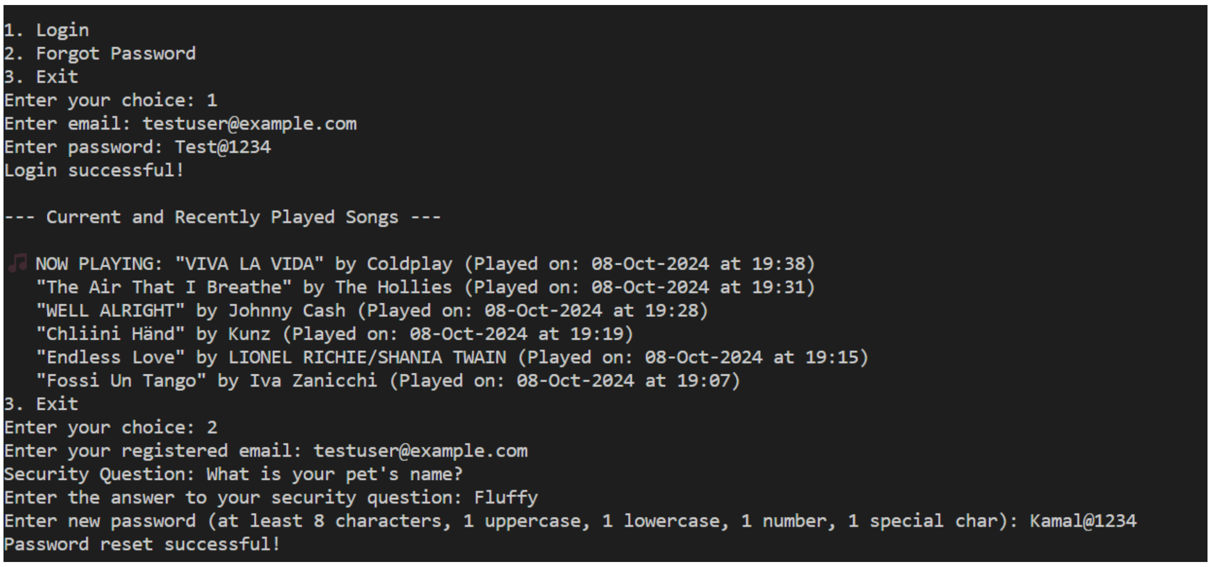
Problem Statement:- Create a Python-based console application that allows users to securely log in and retrieve the latest song list playing on a specific radio station channel using the SRF Song List API. The application will include a secure login system, a password recovery option, and a limit on login attempts. After a successful login, users will be able to view the current playlist from the specified radio station.

```
import csv

import re

import bcrypt

import requests


# Constants for file and API

USERS_FILE = 'users.csv'

API_URL =
'https://il.srgssr.ch/integrationlayer/2.0/srf/songList/radio/byChannel/69e8ac1
6-4327-4af4-b873-fd5cd6e895a7'


# Function to hash password

def hash_password(password):

   return bcrypt.hashpw(password.encode('utf-8'),
bcrypt.gensalt()).decode('utf-8')
```

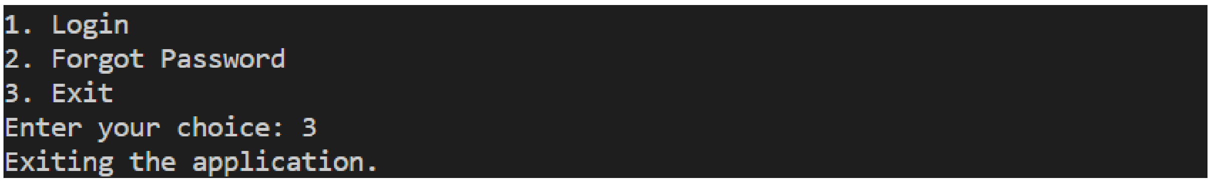
```
# Function to validate email format
def validate_email(email):
    return re.match(r"[^@]+@[^@]+\.[^@]+", email)

# Function to validate password complexity
def validate_password(password):
    return (len(password) >= 8 and
            re.search(r'[A-Z]', password) and
            re.search(r'[a-z]', password) and
            re.search(r'\d', password) and
            re.search(r'[@$!%*?&]', password))

# Function to check login attempts
def check_login_attempts(attempts):
    if attempts >= 5:
        print("You have been locked out due to too many failed login attempts.")
        exit()

# Function to reset password
def reset_password(email, security_answer):
    with open(USERS_FILE, 'r') as file:
        rows = list(csv.DictReader(file))

    for row in rows:
        if row['email'] == email and row['security_answer'] == security_answer:
            new_password = input("Enter new password (at least 8 characters, 1
uppercase, 1 lowercase, 1 number, 1 special char): ")
```

```
            if not validate_password(new_password):
                print("Invalid password format. Try again.")
                return
            row['hashed_password'] = hash_password(new_password)
            break
    else:
        print("Invalid email or security answer.")
        return

    with open(USERS_FILE, 'w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=['email', 'hashed_password',
'security_question', 'security_answer'])
        writer.writeheader()
        writer.writerows(rows)
    print("Password reset successful!")

# Function to fetch the song list from the SRF API
from datetime import datetime

def format_datetime(date_string):
    # Convert ISO datetime string to a more readable format
    date_obj = datetime.fromisoformat(date_string[:-6])  # Removing the
timezone part
    return date_obj.strftime("%d-%b-%Y at %H:%M")

def fetch_song_list():
    response = requests.get(API_URL)
    if response.status_code == 200:
        try:
            songs = response.json()

            if 'songList' in songs and len(songs['songList']) > 0:
                print("\n--- Current and Recently Played Songs ---\n")

                for song in songs['songList']:
                    title = song.get('title', 'Unknown Title')
                    artist = song['artist'].get('name', 'Unknown Artist')
                    date = format_datetime(song.get('date', 'Unknown Date'))
                    is_playing_now = song.get('isPlayingNow', False)

                    if is_playing_now:
                        print(f"🎵 NOW PLAYING: \"{title}\" by {artist} (Played on:
{date})")
                    else:
                        print(f"   \"{title}\" by {artist} (Played on: {date})")
            else:
                print("No songs found in the song list.")

        except ValueError:
            print("Error: Could not parse the response as JSON.")

    else:
        print(f"Failed to fetch the song list. HTTP Status Code:
{response.status_code}")


# Function to handle login process
def login():
    attempts = 0
    while attempts < 5:
        email = input("Enter email: ")
        if not validate_email(email):
            print("Invalid email format.")
            continue

        password = input("Enter password: ")
        if not validate_password(password):
            print("Invalid password format.")
            continue

        with open(USERS_FILE, 'r') as file:
            reader = csv.DictReader(file)
            for row in reader:
                if row['email'] == email and bcrypt.checkpw(password.encode('utf-8'),
row['hashed_password'].encode('utf-8')):
                    print("Login successful!")
                    fetch_song_list()
                    return
            print("Invalid email or password.")
            attempts += 1
            check_login_attempts(attempts)

    print("Maximum login attempts exceeded. Please try again later.")

# Function for the forgot password process
def forgot_password():
    email = input("Enter your registered email: ")
    with open(USERS_FILE, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['email'] == email:
                print(f"Security Question: {row['security_question']}")
                security_answer = input("Enter the answer to your security question:
")
                reset_password(email, security_answer)
                return
    print("Email not found.")

# Main menu
def main_menu():
    while True:
        print("\n1. Login")
        print("2. Forgot Password")
        print("3. Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            login()
        elif choice == '2':
            forgot_password()
        elif choice == '3':
            print("Exiting the application.")
            break
        else:
            print("Invalid choice. Try again.")

# Initial user creation (for testing, remove/comment out in production)
def create_test_user():
    email = "testuser@example.com"
    password = "Test@1234"
    hashed_password = hash_password(password)
    security_question = "What is your pet's name?"
    security_answer = "Fluffy"

    with open(USERS_FILE, 'w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=['email', 'hashed_password',
'security_question', 'security_answer'])
        writer.writeheader()
        writer.writerow({
            'email': email,
            'hashed_password': hashed_password,
            'security_question': security_question,
            'security_answer': security_answer
        })
```

```
if __name__ == '__main__':
    # Uncomment the following line to create a test user (only for initial setup)
    create_test_user()

    # Start the main menu
    main_menu()
```



```
1. Login
2. Forgot Password
3. Exit
Enter your choice: 1
Enter email: testuser@example.com
Enter password: Test@1234
Login successful!

--- Current and Recently Played Songs ---

🎵 NOW PLAYING: "VIVA LA VIDA" by Coldplay (Played on: 08-Oct-2024 at 19:38)
   "The Air That I Breathe" by The Hollies (Played on: 08-Oct-2024 at 19:31)
   "WELL ALRIGHT" by Johnny Cash (Played on: 08-Oct-2024 at 19:28)
   "Chliini Händ" by Kunz (Played on: 08-Oct-2024 at 19:19)
   "Endless Love" by LIONEL RICHIE/SHANIA TWAIN (Played on: 08-Oct-2024 at 19:15)
   "Fossi Un Tango" by Iva Zanicchi (Played on: 08-Oct-2024 at 19:07)
3. Exit
Enter your choice: 2
Enter your registered email: testuser@example.com
Security Question: What is your pet's name?
Enter the answer to your security question: Fluffy
Enter new password (at least 8 characters, 1 uppercase, 1 lowercase, 1 number, 1 special char): Kamal@1234
Password reset successful!
```



```
1. Login
2. Forgot Password
3. Exit
Enter your choice: 3
Exiting the application.
```