Project Overview: Music Management Application

Welcome to the exciting world of music applications! In this project, you'll build components of a Music Management Application, much like Spotify, to help users manage their music effortlessly. You'll learn to create user-friendly systems for account sign-ups, secure logins, and easy song searches. You'll also develop features to efficiently manage song data, making it simple for users to find their favourite tracks. This project offers practical coding challenges that will boost your programming skills and give you a taste of real-world application development. Let's dive in and make some musical magic together!

First, you'll create a User Authentication System, learning how to manage user registration and login processes. This ensures secure access for every user to their music collection.	Next, you'll focus on Song Data Management. You'll learn how to organize and retrieve song information, equipping developers with tools to keep everything in order. This will deepen your understanding of music data and its structure.	Finally, you'll build a Songs Management System where users can easily search for songs by title or artist. This will make finding their favourites a breeze, allowing them to explore and enjoy their music effortlessly!
		
User Authentication System	**Songs Data Management**	**Songs Management System**

Project Instructions (execution and deliverables)

- This project requires you to work in a Team, thus it's essential to use GitHub for collaboration, version control and project management
- **Project deliverables :**
  - Code files
    - Submit a total of three Python Scripts (.py files) one for each problem.
    - These scripts should be fully executable (independently) when run in visual studio code or Spyder IDE.
    - Ensure your code is well-commented to follow best practices for clarity and maintainability
  - User guide
    - Provide a brief document that serves as a user guide for your codes
    - The guide should include clear, step-by-step instructions on how to execute the scripts, including any input dependencies (e.g., files) needed for testing.
  - GitHub Repository for the project
    - Create a GitHub repository for the project and ensure that all team members contribute. Each team member's contribution via GitHub will also be reviewed, so ensure everyone actively participates.
    - Use GitHub for commit history, branching, and collaborating on the project.
- You are free to choose either Functional Programming, OOP, or a combination of both to solve each problem. Ensure your approach is clear and well-organized.
- Your code will be tested on multiple versions of input dictionaries or text files with the same format, so make sure your logic can handle variations effectively.

Problem 01 : User Authentication System

Imagine you are building part of a Music Management Application (like Spotify) where users need to either sign up or log in to access their music and playlists. Your task is to create a **User Authentication System** that handles the registration and login process for users.

The program you write should execute through a menu with the following options:

1. **Signup** – Collect details such as email ID, password, first name, last name, and age. After registering, this information should be stored securely in a dictionary “user_data” that holds all user data.
2. **Sign-in** – Validate the user's credentials (email ID and password) and allow them to log in if the details match the ones stored in the system.
3. **Exit** – Provide an option for users to exit the application.

The user_data dictionary will already contain some pre-populated users so you can test the Sign-in functionality without creating new accounts from scratch.

Tasks under Problem 01 (User Authentication System)

- **User Signup**:
  - ✓ When registering a new user, the program should prompt for:
    - Email ID
    - Password
    - First Name
    - Last Name
    - Age
  - ✓ The program should store all these details in a **dictionary** 'user_data' so that each user's information is associated with their email ID.
  - ✓ If an email ID already exists in the system, the program should notify the user and ask for a different one.
- **User Sign-in**:
  - ✓ The program should allow an existing user to log in by entering their email ID and password.
  - ✓ If the login credentials are correct, the program should welcome the user by name.
  - ✓ If the credentials are incorrect, it should give an error message and prompt them to try again.
- **Exit the Program**
  - ✓ Provide an option for the user to exit the application gracefully.

Sample Dictionary : user_data

```
user_data = {
    "john.doe@example.com": {"password": "john123", "first_name": "John", "last_name": "Doe", "age": 28},
    "jane.smith@example.com": {"password": "jane456", "first_name": "Jane", "last_name": "Smith", "age": 32},
    "alice.jones@example.com": {"password": "alice789", "first_name": "Alice", "last_name": "Jones", "age": 24},
    "bob.brown@example.com": {"password": "bob101", "first_name": "Bob", "last_name": "Brown", "age": 30},
    "charlie.white@example.com": {"password": "charlie202", "first_name": "Charlie", "last_name": "White", "age": 35},
    "diana.green@example.com": {"password": "diana303", "first_name": "Diana", "last_name": "Green", "age": 27},
    "evan.black@example.com": {"password": "evan404", "first_name": "Evan", "last_name": "Black", "age": 29},
    "fiona.red@example.com": {"password": "fiona505", "first_name": "Fiona", "last_name": "Red", "age": 22},
    "george.blue@example.com": {"password": "george606", "first_name": "George", "last_name": "Blue", "age": 26},
    "hannah.yellow@example.com": {"password": "hannah707", "first_name": "Hannah", "last_name": "Yellow", "age": 31}
}
```

- This is provided for reference
- Create your own dictionary and use as a starting point

Expected Output Samples for Problem 01

Use these outputs as reference to understand how the program should behave when tested

```
Menu:
1. Signup
2. Sign-in
3. Exit

Choose an option (1, 2, or 3): 1

Enter your email ID: user1@gmail.com

Enter your password: pass123

Enter your first name: gs

Enter your last name: sg

Enter your age: 32
Registration successful!
```

```
Menu:
1. Signup
2. Sign-in
3. Exit

Choose an option (1, 2, or 3): 1

Enter your email ID: jane.smith@example.com
Email ID already exists. Please use a different one.
```

```
Menu:
1. Signup
2. Sign-in
3. Exit

Choose an option (1, 2, or 3): 2

Enter your email ID: user1@gmail.com

Enter your password: 10293
Invalid email ID or password.
```

```
Menu:
1. Signup
2. Sign-in
3. Exit

Choose an option (1, 2, or 3): 3
Exiting the application. Goodbye!
```

Problem 02 : Songs Management System (Dev)

Imagine you are building a part of a Music Management Application (like Spotify) where developers need to manage song data effectively. Your task is to create functionalities that allow developers to interact with a database of songs efficiently.

The program you write should execute through a menu with the following options:

1. **Load Song Data** - Read song information from a text file and store it in a nested dictionary, with each artist as a key and a list of their songs as the value.
2. **View Songs Database** - Display all songs in the database in a neat format, including title, artist, and genre.
3. **Delete a Particular Song** - Enable the developer to delete a specific song by title from the database.
4. **Modify Song Information** - Allow the developer to update details (album, genre, duration) of a specific song in the database.

Tasks under Problem 02 (Songs Management System - Dev)

- **Load Song Data:**
  - ✓ Prompt the developer to enter the file name containing song information.
  - ✓ Read song information from the specified text file (refer sample file “songs_data.txt”)
  - ✓ Store each song's Title, Artist, Album, Genre, and Duration in a Dictionary, where :
    - Key is Artist names
    - Value is a Lists of songs where each element in the list is a dictionary of song’s attributes
  - ✓ Handle file reading errors gracefully (e.g., file not found error, etc.).
- **View Songs Database:**
  - ✓ Display all songs in the database.
  - ✓ Show details: Title, Artist, and Genre.
  - ✓ Format the output neatly for easy reading.
- **Delete a Particular Song:**
  - ✓ Prompt the developer to enter the artist and title of the song to delete.
  - ✓ Remove the specified song from the database.
  - ✓ Inform the developer if the song was successfully deleted or if it was not found.
- **Modify Song Information:**
  - ✓ Prompt the developer to enter the artist and title of the song to modify.
  - ✓ Prompt for the details to be modified: Album, Genre, or Duration.
  - ✓ Update the song’s information in the database.
  - ✓ Confirm the changes to the developer.

Sample File : songs_data.txt

songs_data - Notepad

File Edit Format View Help

```
"I Will Always Love You","Whitney Houston","The Bodyguard","Soundtrack","4:31"
"Greatest Love of All","Whitney Houston","Whitney","Pop","4:50"
"Billie Jean","Michael Jackson","Thriller","Pop","4:54"
"Black or White","Michael Jackson","Dangerous","Pop","4:15"
"Wonderwall","Oasis","(What's the Story) Morning Glory?","Rock","4:18"
"Champagne Supernova","Oasis","(What's the Story) Morning Glory?","Rock","7:00"
"Smells Like Teen Spirit","Nirvana","Nevermind","Rock","5:01"
"Come As You Are","Nirvana","Nevermind","Rock","3:38"
"Torn","Natalie Imbruglia","Left of the Middle","Pop","4:05"
"Shimmer","Fuel","Something Like Human","Rock","3:40"
"Backstreet's Back","Backstreet Boys","Backstreet Boys","Pop","3:35"
"I Want It That Way","Backstreet Boys","Millennium","Pop","3:33"
"My Heart Will Go On","Celine Dion","Let's Talk About Love","Pop","4:40"
"Because You Loved Me","Celine Dion","Falling Into You","Pop","4:34"
"Under the Bridge","Red Hot Chili Peppers","Blood Sugar Sex Magik","Rock","4:24"
"Scar Tissue","Red Hot Chili Peppers","California","Rock","3:37"
```

- This view is provided just for reference
- Create your own file

Expected Output Samples for Problem 02 (1/2)

```
Developer Menu:
1. Load Song Data
2. View Songs Database
3. Delete a Song
4. Modify a Song
5. Exit

Select an option: 1

Enter the file name to load songs: songs_data.txt
Songs loaded from songs_data.txt.
```

Use these outputs as reference to understand how the program should behave when tested

```
Developer Menu:
1. Load Song Data
2. View Songs Database
3. Delete a Song
4. Modify a Song
5. Exit

Select an option: 2

Songs Database:
Title                          Artist                      Genre
================================================================
"I Will Always Love You"       "Whitney Houston"           "Soundtrack"
"Greatest Love of All"         "Whitney Houston"           "Pop"
"Billie Jean"                  "Michael Jackson"           "Pop"
"Black or White"               "Michael Jackson"           "Pop"
"Wonderwall"                   "Oasis"                     "Rock"
"Champagne Supernova"          "Oasis"                     "Rock"
"Smells Like Teen Spirit"      "Nirvana"                   "Rock"
"Come As You Are"              "Nirvana"                   "Rock"
"Torn"                         "Natalie Imbruglia"         "Pop"
"Shimmer"                      "Fuel"                      "Rock"
"Backstreet's Back"            "Backstreet Boys"           "Pop"
"I Want It That Way"           "Backstreet Boys"           "Pop"
"Shape of My Heart"            "Backstreet Boys"           "Pop"
"My Heart Will Go On"          "Celine Dion"               "Pop"
"Because You Loved Me"         "Celine Dion"               "Pop"
"Under the Bridge"             "Red Hot Chili Peppers"     "Rock"
"Scar Tissue"                  "Red Hot Chili Peppers"     "Rock"
```

Expected Output Samples for Problem 02 (2/2)

```
Select an option: 3

Enter the artist's name of the song to delete: "Bon Jovi"

Enter the title of the song to delete: "Livin' on a Prayer"
Deleted '"Livin' on a Prayer"' by "Bon Jovi" from the database.
```

Use these outputs as reference to understand how the program should behave when tested

```
Select an option: 4

Enter the artist's name of the song to modify: "Oasis"

Enter the title of the song to modify: "Wonderwall"
Current details:
Title: "Wonderwall", Album: "(What's the Story) Morning Glory?",
Genre: "Rock", Duration: "4:18"

Enter new album (or press Enter to keep current):

Enter new genre (or press Enter to keep current): Pop

Enter new duration (or press Enter to keep current):
Modified '"Wonderwall"' by "Oasis".
```

```
Select an option: 2

Songs Database:
Title                          Artist                    Genre
================================================================================
"I Will Always Love You"       "Whitney Houston"         "Soundtrack"
"Greatest Love of All"         "Whitney Houston"         "Pop"
"Billie Jean"                  "Michael Jackson"         "Pop"
"Black or White"               "Michael Jackson"         "Pop"
"Wonderwall"                   "Oasis"                   Pop
"Champagne Supernova"          "Oasis"                   "Rock"
```

Problem 03 : Songs Management System (User)

Imagine you are building a part of a Music Management Application (like Spotify) where users need to search for songs and explore their information. Your task is to create a Songs Management System that allows users to interact with a database of songs efficiently

The program you write should execute through a menu with the following options:

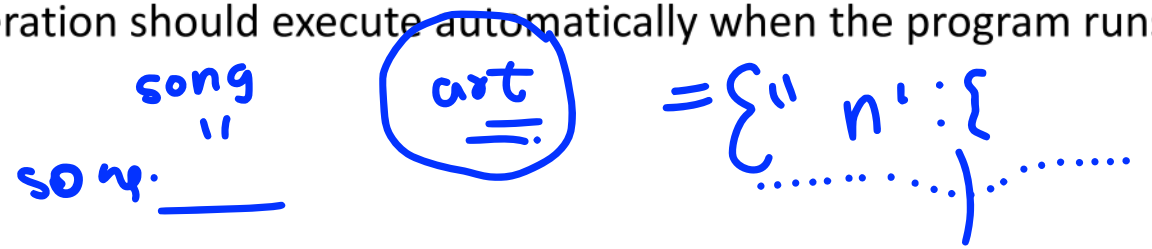
1. **Search for a Song by Title** - Allow the user to input a song title and retrieve its details. If found, display the song's information (Artist, Album, Genre, Duration). If not found, inform the user that the title does not exist.
2. **Search for All Songs by an Artist** - Enable the user to search for all songs by a specific artist. If songs are found, display a list of their details. If no songs are found, notify the user.
3. **Exit the Program** - Provide an option for the user to exit the application gracefully.

Tasks under Problem 03 (Songs Management System - User)

- **Search for a Song by Title:**
  - ✓ Prompt the user to enter a song title.
  - ✓ Search the database for songs that match the input title (case-insensitive).
  - ✓ If found, display the song details (Artist, Album, Genre).
  - ✓ If no match is found, inform the user that the song title does not exist in the database.

- **Search for All Songs by an Artist:**
  - ✓ Prompt the user to input an artist's name.
  - ✓ Search the database for all songs associated with that artist.
  - ✓ If songs are found, display the list of songs with their details (Title, Album, Genre, Duration).
  - ✓ If no songs are associated with the artist, notify the user that the artist does not have songs in the database.

Hint : Consider reading the songs data from a text file "songs_database.txt" file and converting it into a nested dictionary format which can act as your database. This approach will enable efficient searching for the tasks that follow. Also, this operation should execute automatically when the program runs, without requiring any involvement from user.



Sample File : songs_database.txt

songs_database - Notepad

File Edit Format View Help

```
"Shape of You", "Ed Sheeran", "Divide", "Pop", "3:53"
"Blinding Lights", "The Weeknd", "After Hours", "Synth Pop", "3:20"
"Levitating", "Dua Lipa", "Future Nostalgia", "Disco", "3:24"
"Rolling in the Deep", "Adele", "21", "Pop", "3:48"
"Stay", "The Kid LAROI", "FCK LOVE 3", "Hip Hop", "2:21"
"Bad Guy", "Billie Eilish", "When We All Fall Asleep Where Do We Go", "Pop", "3:14"
"Someone Like You", "Adele", "21", "Pop", "4:45"
"Watermelon Sugar", "Harry Styles", "Fine Line", "Pop", "3:03"
"Dance Monkey", "Tones and I", "The Kids Are Coming", "Pop", "3:29"
"Peaches", "Justin Bieber", "Justice", "R&B", "3:18"
"Good 4 U", "Olivia Rodrigo", "Sour", "Pop", "2:58"
"drivers license", "Olivia Rodrigo", "Sour", "Pop", "4:02"
"Dynamite", "BTS", "BE", "K Pop", "3:19"
"Dont Start Now", "Dua Lipa", "Future Nostalgia", "Disco", "3:03"
"Senorita", "Shawn Mendes and Camila Cabello", "Senorita", "Pop", "3:11"
"Perfect", "Ed Sheeran", "Divide", "Pop", "4:23"
"Shallow", "Lady Gaga and Bradley Cooper", "A Star is Born", "Pop", "3:37"
"I Will Always Love You", "Whitney Houston", "The Bodyguard", "Pop", "4:31"
"Uptown Funk", "Mark Ronson ft Bruno Mars", "Uptown Special", "Funk", "4:30"
"Lose Yourself", "Eminem", "8 Mile", "Hip Hop", "5:26"
"Thinking Out Loud", "Ed Sheeran", "Multiply", "Pop", "4:41"
"Viva La Vida", "Coldplay", "Viva La Vida or Death and All His Friends", "Rock", "4:02"
"My Heart Will Go On", "Celine Dion", "Titanic", "Pop", "4:40"
```

- This view is provided just for reference
- Create your own file

Expected Output Samples for Problem 03

```
--- User Menu ---
1. Search for a Song by Title
2. Search for All Songs by an Artist
3. Exit

Select an option: 1

Enter the song title to search: Uptown Funk

Searching for songs with title: 'Uptown Funk'
Found: 'Uptown Funk' by Mark Ronson ft Bruno Mars (Genre: Funk)
```

Use these outputs as reference to understand how the program should behave when tested

```
--- User Menu ---
1. Search for a Song by Title
2. Search for All Songs by an Artist
3. Exit

Select an option: 2

Enter the artist's name to search: Ed Sheeran

Searching for songs by artist: 'Ed Sheeran'
Found: 'Shape of You' (Album: Divide, Genre: Pop, Duration: 3:53)
Found: 'Perfect' (Album: Divide, Genre: Pop, Duration: 4:23)
Found: 'Thinking Out Loud' (Album: Multiply, Genre: Pop, Duration:
4:41)
```

```
--- User Menu ---
1. Search for a Song by Title
2. Search for All Songs by an Artist
3. Exit

Select an option: 3
Exiting the Songs Management System. Goodbye!
```

End of Document