Francesco Saverio Serpico **Matricola n°:** N45007430

REPORT

Per avviare il programma, seguire questi passaggi:

1. ***Assicurarsi di avere tutti i file necessari per il progetto.***
2. ***Aprire il terminale e spostarsi nella directory del progetto.***
3. ***Compilare il codice sorgente utilizzando il comando "make".***
4. ***Una volta che il codice è stato compilato, eseguire il programma con il comando "./netflix".***
5. ***Seguire le istruzioni del menu per utilizzare il programma.***
6. ***In caso di uscita, ripulire con ”make fclean” e tornare al passaggio 3.***

La traccia chiede di creare un programma per la gestione di un database di account utente, ognuno dei quali può avere fino a quattro profili. I profili possono essere utilizzati per tenere traccia delle preferenze di visualizzazione di un utente per un determinato servizio.

Per implementare queste funzionalità, abbiamo deciso di creare le seguenti classi:

Account: rappresenta un account di un utente, con un indirizzo email e una password, e un elenco di profili associati all'account.

Profilo: rappresenta un profilo di un membro della famiglia di un utente, con un nome e un'età.

Database: rappresenta il database di film e serie TV disponibili per la visualizzazione, con un elenco di film e serie TV.

Film: rappresenta un film, con un titolo, un genere, una durata e un elenco di voti.

Serie: rappresenta una serie TV, con un titolo, un genere, un numero di stagioni e un elenco di voti.

Il main() creerà un'istanza di Database e di Account, e chiamerà i metodi appropriati per gestire le diverse azioni dell'utente (ad esempio, la creazione di un nuovo profilo, la ricerca di film o serie TV, l'aggiunta di un voto a un film o serie TV).

Abbiamo deciso di utilizzare le std::vector per memorizzare gli elementi del database e i profili associati a un account poiché queste strutture dati sono flessibili e consentono di aggiungere e rimuovere elementi in modo efficiente. Inoltre, abbiamo deciso di utilizzare std::string per memorizzare i titoli, i generi e gli indirizzi email poiché sono facili da usare e gestire.

Per implementare le funzionalità richieste, abbiamo implementato i seguenti metodi per la classe **Account**:

```
+--------------------------------------+
|          Account                     |
+--------------------------------------+
| - email: std::string                 |
| - password: std::string              |
| - profiles: std::vector<Profilo>     |
| + Account(email: std::string,        |
|           password: std::string)     |
| + addProfile(profile: Profilo)       |
| + setActiveProfile(profile: Profilo) |
| + getNumberOfProfiles(): int         |
+--------------------------------------+
```

- Account(std::string email, std::string password): costruttore che inizializza l'email e la password dell'Account.
- void addProfile(Profilo profile): metodo che permette di aggiungere un nuovo profilo all'Account.
- void setActiveProfile(Profilo profile): metodo che imposta il profilo attivo dell'Account.
- int getNumberOfProfiles(): metodo che restituisce il numero di profili associati all'Account.

Per la classe **Database** abbiamo implementato i seguenti metodi:

```
+--------------------------------------+
|          Database                    |
+--------------------------------------+
| - films: std::vector<Film>           |
| - series: std::vector<Serie>         |
| + addFilm(film: Film)                |
| + addSerie(serie: Serie)             |
| + removeFilm(film: Film)             |
| + removeSerie(serie: Serie)          |
| + getFilms(): std::vector<Film>      |
| + getSeries(): std::vector<Serie>    |
+--------------------------------------+
```

- void addFilm(Film film): metodo che permette di aggiungere un nuovo Film al Database.
- void addSerie(Serie serie): metodo che permette di aggiungere una nuova Serie al Database.
- void removeFilm(Film film): metodo che permette di rim uovere un Film dal Database.
- void removeSerie(Serie serie): metodo che permette di rimuovere una Serie dal Database.
- std::vector<Film> searchFilms(std::string query): metodo che cerca i Film nel Database in base alla query inserita dall'utente e restituisce un elenco di Film che corrispondono alla query.
- std::vector<Serie> searchSeries(std::string query): metodo che cerca le Serie nel Database in base alla query inserita dall'utente e restituisce un elenco di Serie che corrispondono alla query.

Per la classe **Profilo** abbiamo implementato i seguenti metodi:

```
+----------------------------+
|          Profilo           |
+----------------------------+
| - name: std::string        |
|                            |
| + Profilo(name: std::string|
|               )            |
+----------------------------+
```

- Profilo(std::string name): costruttore che inizializza il nome.
- std::string getName(): metodo che restituisce il nome del Profilo.

Per la classe **Serie** abbiamo implementato i seguenti metodi:

```
+--------------------------------+
|           Serie                |
+--------------------------------+
| - title: std::string           |
| - genre: std::string           |
| - seasons: int                 |
| - ratings: std::vector<double> |
| + Serie(title: std::string,    |
|        genre: std::string,     |
|        seasons: int)           |
```

```
| + addRating(rating: double)      |
| + getAverageRating(): double     |
| + getTitle(): std::string        |
+----------------------------------+
```

- Serie(std::string title, std::string genre, int seasons): costruttore che inizializza il titolo, il genere e il numero di stagioni della Serie.
- void addRating(double rating): metodo che permette di aggiungere un nuovo voto alla Serie.
- double getAverageRating(): metodo che restituisce il voto medio della Serie.
- std::string getTitle(): metodo che restituisce il titolo della Serie.

Per la classe **Film** abbiamo inoltre implementato i seguenti metodi:

```
+----------------------------------+
|           Film                   |
+----------------------------------+
| - title: std::string             |
| - genre: std::string             |
| - duration: int                  |
| - ratings: std::vector<double>   |
| + Film(title: std::string,       |
|        genre: std::string,       |
|        duration: int)            |
| + addRating(rating: double)      |
| + getAverageRating(): double     |
| + getTitle(): std::string        |
+----------------------------------+
```

- Film(std::string title, std::string genre, int duration): costruttore che inizializza il titolo, il genere e la durata del Film.
- void addRating(int rating): metodo che permette di aggiungere un nuovo voto al Film.
- double getAverageRating(): metodo che restituisce la media dei voti del Film.
- std::string getTitle(): metodo che restituisce il titolo del Film.
- std::string getGenre(): metodo che restituisce il genere del Film.
- int getDuration(): metodo che restituisce la durata del Film.

Il programma inizia chiedendo all'utente se vuole creare un nuovo account o effettuare il login con un account esistente. Se l'utente sceglie di effettuare il login, viene chiesta l'email e la password per verificare se esiste un account con quelle credenziali. Se l'account esiste, viene impostato come account attivo e l'utente può accedere alle funzionalità del programma. Se invece l'utente sceglie di creare un nuovo account, viene chiesta l'email, la password e l'username, quindi viene creato un nuovo oggetto Account e viene chiesto all'utente di creare fino a quattro profili per l'account. Una volta creati i profili, l'account viene aggiunto all'elenco degli account e viene impostato come account attivo.

Per trovare un film simile, il programma utilizza il metodo getSimilarFilm della classe Database, che confronta il genere del film di riferimento con quello di tutti gli altri film presenti nel database. Se viene trovato un film con lo stesso genere, viene restituito il primo film trovato. Altrimenti, viene restituito un film vuoto.

Il rating è una valutazione assegnata a un film o a una serie TV da parte dell'utente. Il programma tiene traccia del rating di ogni film o serie TV valutato dall'utente e ne calcola la media, che viene utilizzata per determinare il film o la serie TV più apprezzata al momento (selezione delle opzioni 4 o 5 del menu). Il rating viene assegnato utilizzando il metodo rateFilm o rateSerie della classe Database, a seconda che si stia valutando un film o una serie TV.

Quando l'utente sceglie l'opzione "Cerca serie" o "Cerca film", viene richiesto di inserire il titolo di una serie o di un film. Il programma cerca poi la serie o il film nel database utilizzando il metodo findSerie o findFilm del database. Se la serie o il film viene trovato, viene chiesto all'utente se vuole vederlo o vederla utilizzando i metodi watchSerie o watchFilm dell'oggetto Profilo associato all'account attivo. Se la serie o il film non viene trovato, viene visualizzato un messaggio di errore.

I metodi watchSerie e watchFilm: questi metodi aggiungono la serie o il film alla lista delle serie o dei film visti dall'utente e chiedono un feedback sulla visione. Se la serie o il film è già presente nella lista, viene visualizzato un messaggio di avviso.

In sintesi, abbiamo implementato un programma per la gestione di un database di account utente che permette all'utente di creare nuovi account, effettuare il login con account esistenti e gestire i profili di ogni account. Abbiamo creato le classi Account e Profilo per rappresentare gli oggetti che compongono il database e la classe Database per gestire l'elenco degli account. Abbiamo inoltre implementato la funzione saveAccountsToFile per salvare gli account nel file "accounts.txt" ogni volta che viene effettuata una modifica all'elenco degli account.