

SE 360 - Project Report



Watchverse

GROUP MEMBERS:

Özge BOSTAN (20230601013)

Sude SAVAŞ (20230601060)

1. Project Description Watchverse is a collaborative watchlist management system built upon a Client-Server architecture. It integrates with The Movie Database (TMDB) API to provide users with an extensive library of movies and TV shows. Users can search for content, create their private or public watchlists, and form user groups. With the "Link-Only" feature, users can share their specific watchlists within these groups securely.

2. Requirements Analysis

i. Implemented Features:

- **User Authentication:** A registration and login system is implemented using a MySQL database. It includes security questions for account verification when users forget their password.
- **Search Engine:** Users can perform searches for movies and TV series. The results are parsed and displayed on the GUI (**as shown in Figure 3**).
- **Watchlist Management:** Users can create multiple watchlists with specific visibility settings (Private, Public, Link-Only). Items in watchlists include metadata such as titles, genres, and poster images.
- **Group System & Sharing:** Users can create groups and invite others via unique join codes. The core feature allows users to link their "Link-Only" lists to these groups.
- **GUI:** The interface is built using Java Swing, using CardLayout for smooth transitions between panels (see Figure 2).

ii. Future Improvements (Not Done):

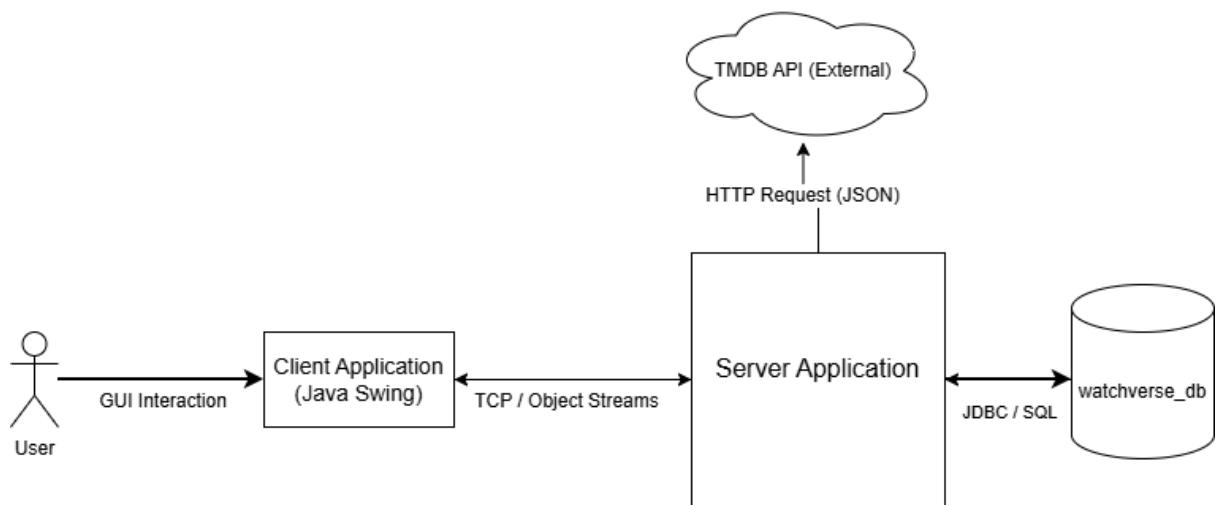
- **Real-time Chat:** While groups exist, a real-time messaging feature for members is currently not implemented.
- **Password Reset via Email:** The system currently relies on security questions rather than email verification for forgotten passwords.
- **Profile Customization:** Users cannot currently upload custom avatars; the system generates default avatars based on initials.

3. System Design & Architecture

Architecture: The system is designed using a multi-threaded Client-Server model.

- **Server:** The server ([Server.java](#)) listens on port. It creates a new `ClientHandler` thread for each connected user. It manages all database transactions via Data Access Objects (DAOs) located in the `Services` and `DataBase` packages.
- **Client:** The client communicates with the server via `ObjectOutputStream` and `ObjectInputStream` to send commands (e.g., `LOGIN`, `SEARCH`) and receive data objects.

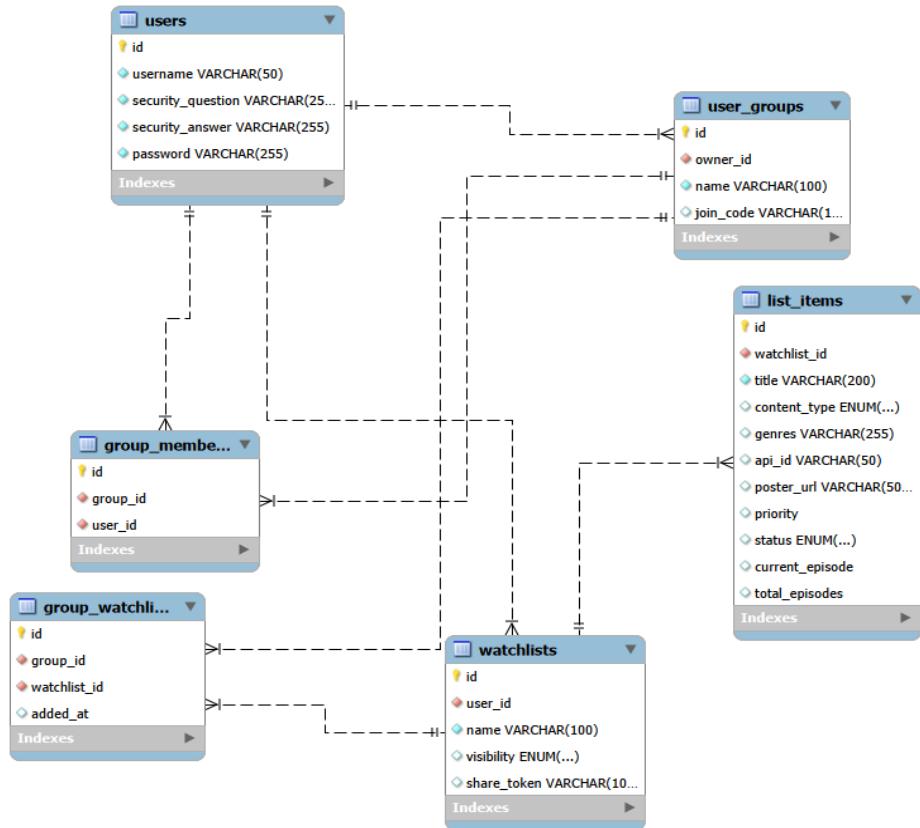
A simple representation of the high-level architecture is shown below:



Database Design: Database of the project managed by [DataBaseManager.java](#). The schema consists of six tables.

1. `users`: Stores authentication credentials.
2. `watchlists`: Stores info about watchlist and visibility settings.
3. `list_items`: Stores the actual movies/series content.
4. `user_groups`: Stores group information and join codes.
5. `group_members`: Manages the many-to-many relationship between users and groups.
6. `group_watchlists`: Links specific watchlists to groups for sharing.

The Entity-Relationship (ER) diagram of the database is provided below:



User Interface: The interface is designed to be user-friendly. The navigation flow starts with the Welcome screen and the Authentication screens.

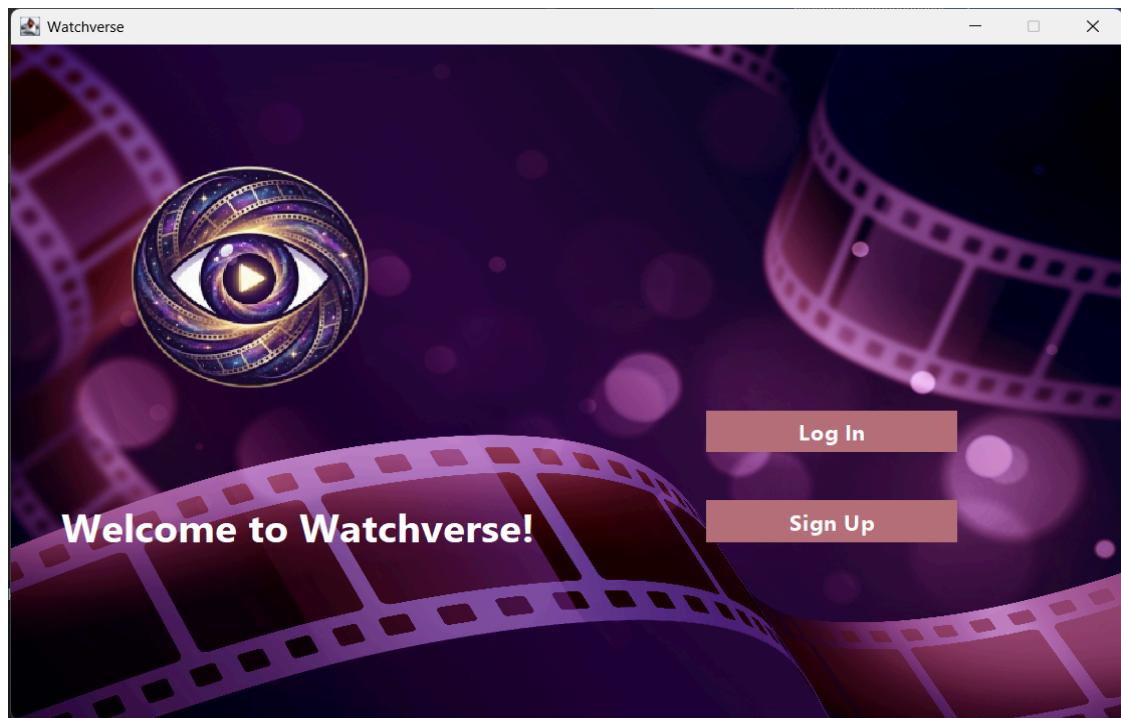


Figure 1: The Welcome Panel allowing users to Login or Register.

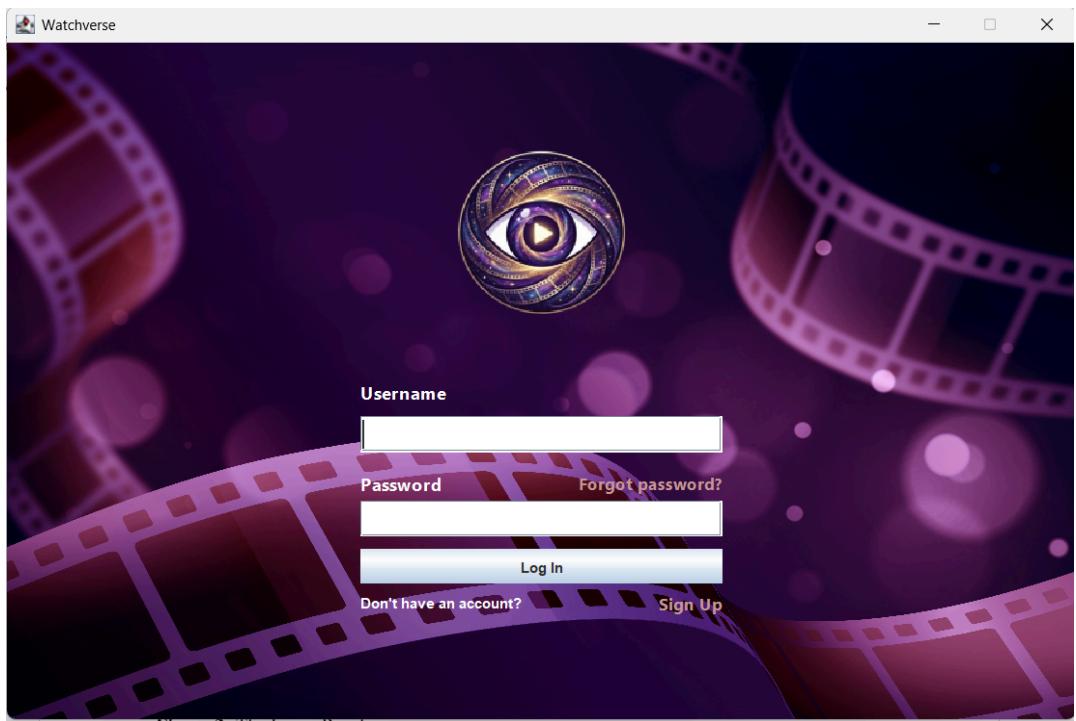


Figure 2: The Login Panel.

Figure 3 shows the main dashboard where users interact with their personal watchlists after a successful login.

A screenshot of the Watchverse dashboard. The top navigation bar shows the title 'Watchverse'. The main area starts with a 'Welcome Ozge0010' message. Below it is a search bar with the placeholder 'Search movies or shows...'. To the right is a '0' icon. Underneath the welcome message are sections for 'My Watchlists' (which shows a placeholder image and a '+ Add' button) and 'My Groups' (also with a placeholder image and a '+ Add' button). To the right of these is a 'Test' section showing 'Public' items with a count of '0 Items'. At the bottom left is a 'Discover Watchlists' section with a placeholder image and a '+ Add' button. A small sidebar on the left lists 'Discover other people's watchlists...' and shows entries like 'Test' and 'PublicDe'.

Figure 3: The Watchverse Dashboard.

4. Java Technologies Used

- **Java SE:** Used for core logic, collections, and IO streams.
- **Java Swing & AWT:** Used for building the Graphical User Interface (Frames, Panels, Layout Managers).
- **Java Networking (java.net):** Used [Socket](#) and [ServerSocket](#) classes to establish TCP communication; and [HttpURLConnection](#) for connecting to the TMDB API.
- **JDBC:** Used [PreparedStatement](#) and [ResultSet](#) for executing secure SQL queries and managing the MySQL database connection.
- **Java Concurrency:** Used [Runnable](#) interface in [ClientHandler](#) to handle multiple clients simultaneously.
- **External APIs:** The Movie Database (TMDB) API is used to fetch real-time metadata.
- **Data Parsing:** The [org.json](#) library was used to parse the JSON responses received from the API.

5. User Manual Pre-requisites (Before Running):

1. **External Library Requirement:** This project uses the [org.json](#) library to parse API responses. Please add the [org.json](#) library to the project's classpath before compiling.

Database Initialization: Although [DataBaseManager](#) includes an auto-initialization, permission settings on different machines may prevent automatic creation. To ensure the application runs correctly, please execute the following SQL commands.

CREATE DATABASE IF NOT EXISTS watchverse_db;

USE watchverse_db;

2. *Note: Open the [src/resources/config.properties](#) file and update the db.user and db.password fields according to your local MySQL configuration.*

Steps to Use:

1. **Starting the Application:** Run [Watchverse.java](#). The system will automatically create the tables if they do not exist and launch the Client and the Server.
2. **Registration/Login:** Register using a unique username and password. After creating an account, you can log in.

3. **Creating a List:** On the dashboard, click the "+" button next to "My Watchlists". Name your list and select visibility (Choose "Link-Only" if you intend to share it within groups).
4. **Searching & Adding:** Type a movie name in the top search bar and press Enter. Click on a movie/series poster to add it to a selected list. Ensure a watchlist is selected before adding. (see **Figure 4**).

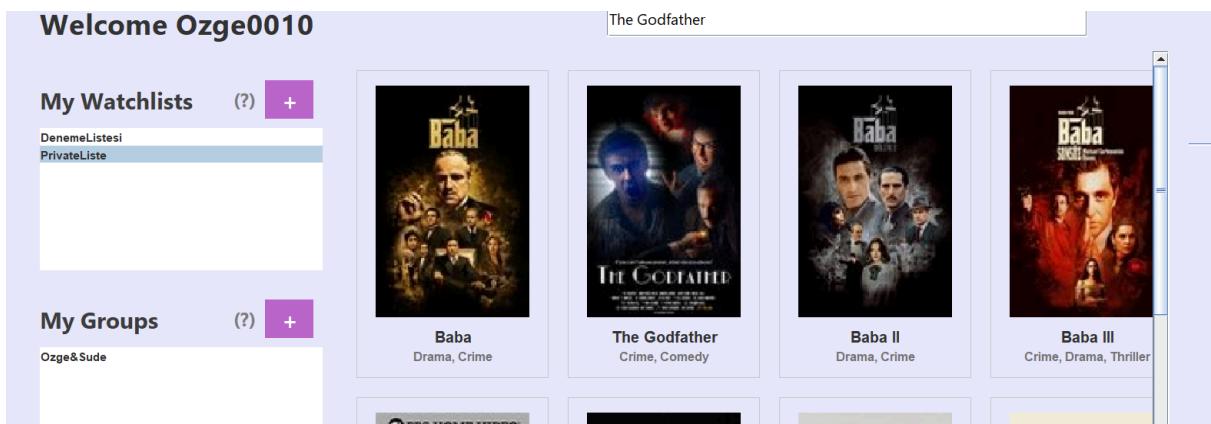
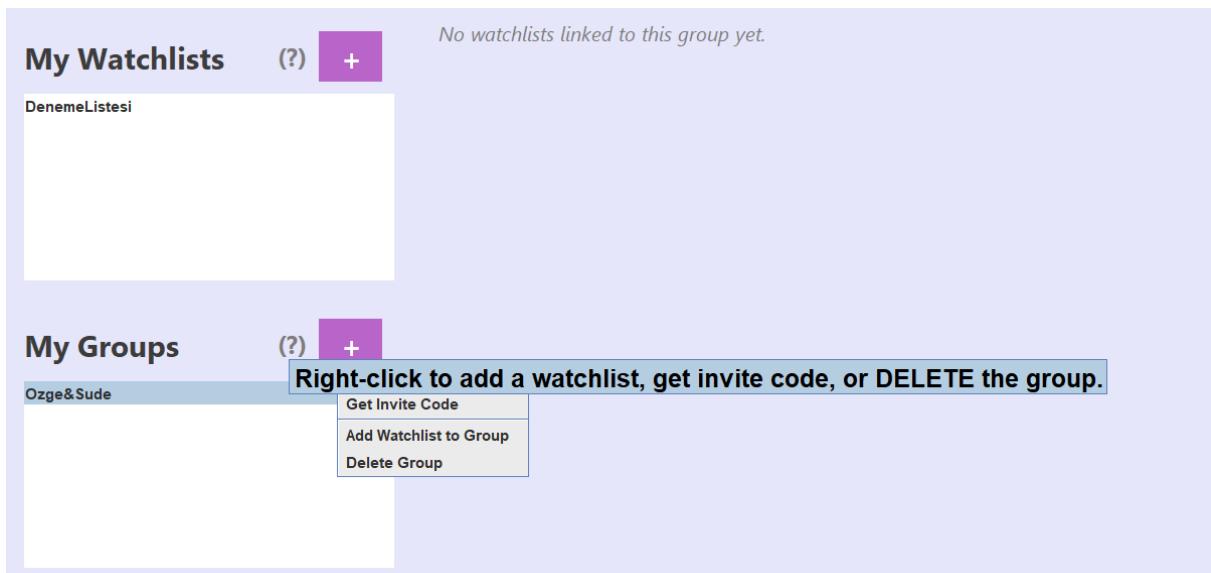


Figure 4: Adding movie/series to the watchlist

5. **Groups & Sharing:** Create a group via the "My Groups" panel. Right-click on your group name and select "Add Watchlist". Choose one of your Link-Only lists to share it with the group (see **Figure 5**).



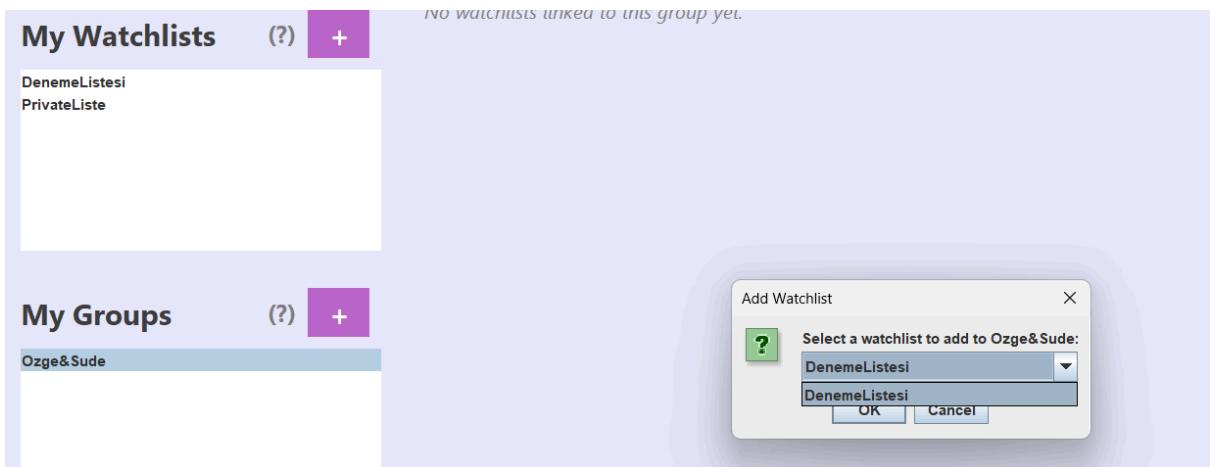


Figure 5: Linking watchlists to groups via the context menu.

6. **Viewing Groups:** Once a list is linked, group members can click on the group name to see shared lists (see **Figure 6**).

The screenshot shows the 'Group View' interface. At the top, it says 'Welcome Ozge0010'. Below this are two main sections: 'My Watchlists' and 'My Groups'. The 'My Watchlists' section displays the 'DenemeListesi' watchlist. The 'My Groups' section displays the 'Ozge&Sude' group. To the right of the 'My Watchlists' section, there is a preview box showing the details of the 'DenemeListesi' watchlist, which was added earlier.

Figure 6: The Group View displaying shared watchlists.

7. **Discovering Public Watchlists:** Locate the search bar at the bottom-left corner labeled "Discover other people's watchlists...". Type keywords to filter and find public lists created by other users globally.
8. **Account Management:** To change your password or delete your account, click on your profile icon (top-right corner) and select "Settings". This opens a dialog where you can update your credentials or perform account deletion actions (**see Figure 7**).

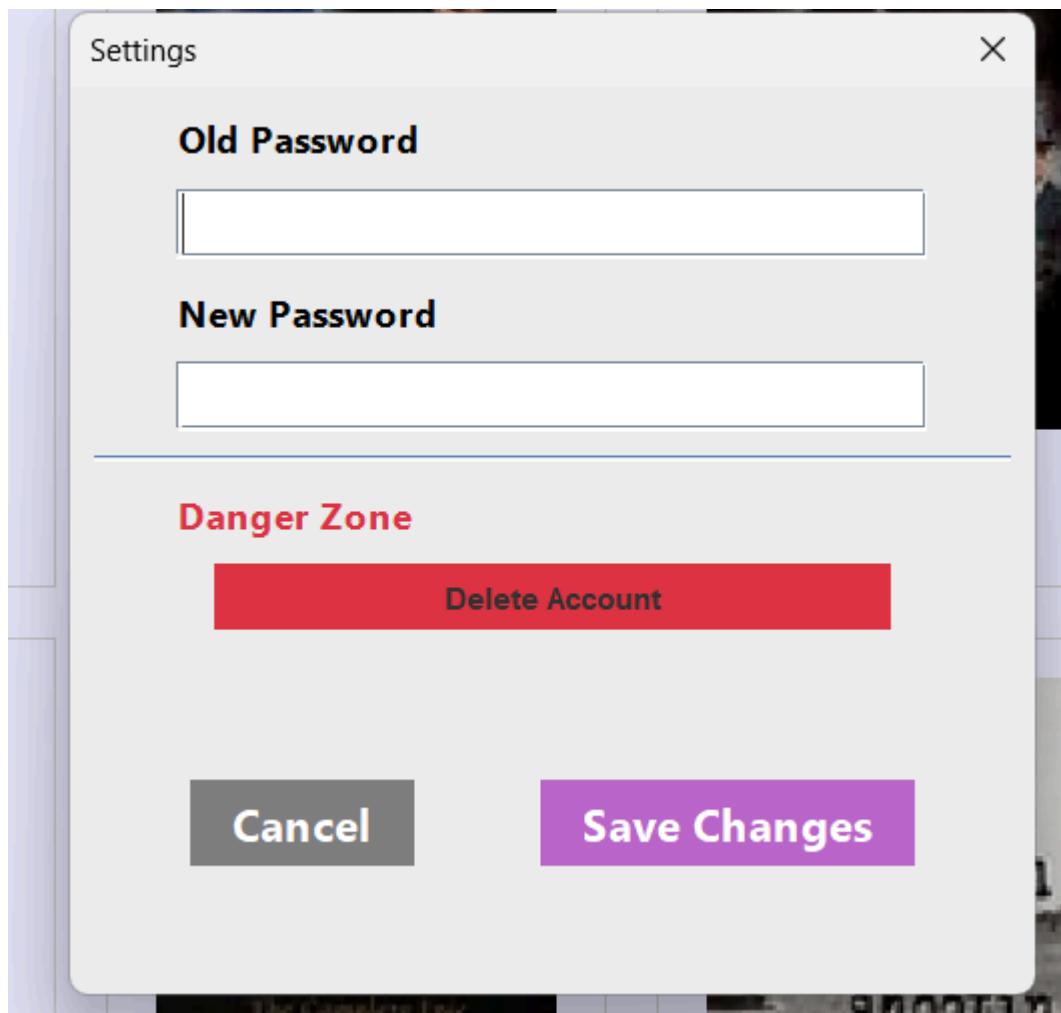


Figure 7: The Settings menu.

6. **AI/Tool Usage Declaration** Gemini AI was used to debug socket communication errors and optimize the SQL queries (specifically JOIN operations in `WatchlistDao`) for the group sharing logic.