# **Movie Randomizer Database Overview**

**The database contains 3 core tables:**

* **users** – stores registered user information and preferences
* **movies** – holds all movie records
* **collections** – links users to their saved movies (many-to-many relationship)

## 📋 Table Breakdown

### **1. users**

Stores account details and personal movie preferences.

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT, PK | Unique auto-increment user ID |
| username | VARCHAR(50) | Unique username |
| email | VARCHAR(100) | Unique email address |
| password\_hash | VARCHAR(255) | Hashed password using password\_hash() in PHP |
| genre | VARCHAR(50) | User’s preferred movie genre (e.g., Drama, Action) |
| mood | VARCHAR(50) | User’s preferred mood (e.g., Chill, Tense) |
| min\_runtime | INT | Minimum preferred movie runtime |
| max\_runtime | INT | Maximum preferred movie runtime |

✅ All fields are required except runtime preferences (can be null if unfiltered).  
✅ Enforced uniqueness on username and email.

### **2. movies**

Holds all movies in the system.

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT, PK | Unique movie ID |
| title | VARCHAR(150) | Movie title |
| genre | VARCHAR(50) | Genre category |
| mood | VARCHAR(50) | Associated emotional tone (e.g., Tense) |
| runtime | INT | Runtime in minutes |
| poster\_url | VARCHAR(255) | Optional: URL for the movie poster image |

✅ This table is used in filtering (via genre/mood/runtime) for the randomizer.

### 3. collections

Join table connecting users and movies (many-to-many relationship).

| **Column** | **Type** | **Description** |
| --- | --- | --- |
| id | INT, PK | Unique ID for each collection entry |
| user\_id | INT, FK | References users(id) |
| movie\_id | INT, FK | References movies(id) |
| added\_on | DATETIME | Timestamp of when the movie was added |

✅ FOREIGN KEY constraints include ON DELETE CASCADE  
✅ Prevents orphan entries if a user or movie is deleted

## 🔗 Relationships

* **One user → many collection entries**
* **One movie → can appear in many collections**
* This is a **many-to-many relationship** modeled through the collections table

## 🧪 Sample Usage

* spin\_wheel.php: Filters movies using users.genre, users.mood, users.min\_runtime, etc.
* collection\_get.php: Uses JOIN to pull all movies in a user's collection
* collection\_add.php: Inserts a new row into collections
* collection\_remove.php: Deletes one row from collections based on user\_id and movie\_id