Project Requirements

1. Movie Model

Each movie should be represented as an object with the following properties:

- `id` (number): A unique identifier for each movie.
- `title` (string): The title of the movie.
- `genre` (string): The genre of the movie (Action, Comedy, Drama, Fantasy, Horror, Sci-Fi, Thriller, Animation).
- `year` (number): The release year of the movie.
- `rating` (number): The rating of the movie (e.g., 1 to 10).
- `watched` (boolean): A flag indicating whether the movie has been watched.

2. Movie Management

The application should support the following operations. Each operation identifies a movie by its ID:

- Add a New Movie: Users can add a new movie by providing the title, genre, year, and rating. The movie should be assigned a unique ID automatically.
- Edit a Movie: Users can update the details of an existing movie (title, genre, year, and rating).
- Toggle Watched Status: Users can toggle the watched status of a movie (mark it as watched or unwatched).
- Remove a Movie: Users can delete a movie from the watchlist.

3. Watchlist Display

The application should include functionality for:

- Find a Movie: Users can search for a movie by its title and display its details.
- View All Movies: Users can view all movies currently in the watchlist.
- Filter Movies: Users can filter movies based on their watched status. Options should include:
    - All: Show all movies.
    - Watched: Show only movies that have been marked as watched.
    - Unwatched: Show only movies that have not been watched.
- Sort Movies: Users can sort movies by different criteria. Sorting options should include:
    - Title: Alphabetically by movie title.
    - Genre: Alphabetically by genre.
    - Year: By release year, ascending .
    - Rating: By rating, descending .

4. Code Structure and Design

- Modularity: Functions should be modular and handle specific operations like adding, editing, marking as watched, and removing movies. Each function should perform a single, well-defined task.
- Data Integrity: Ensure that movie data is managed consistently, with unique IDs assigned to each movie and accurate tracking of the watched status.

- **Error Handling**: Include appropriate error handling to manage cases where operations are attempted on non-existent movies or invalid input.
- **Seek Help**: Utilize online resources or reference example solutions if you encounter challenges.

5. Testing

Here is a test movie set that you can use for testing your movie watchlist application:

```javascript
const movieWatchlist = [
    { id: 1, title: "Parasite", genre: "Thriller", year: 2019, rating: 8.6, watched: true },
    { id: 2, title: "Spirited Away", genre: "Animation", year: 2001, rating: 8.6, watched: false },
    { id: 6, title: "Interstellar", genre: "Sci-Fi", year: 2014, rating: 8.6, watched: false },
    { id: 7, title: "Get Out", genre: "Horror", year: 2017, rating: 7.7, watched: true },
    { id: 8, title: "Coco", genre: "Animation", year: 2017, rating: 8.4, watched: true },
    { id: 9, title: "La La Land", genre: "Musical", year: 2016, rating: 8.0, watched: false },
    { id: 11, title: "Arrival", genre: "Sci-Fi", year: 2016, rating: 7.9, watched: false },
    { id: 12, title: "Whiplash", genre: "Drama", year: 2014, rating: 8.5, watched: true },
    { id: 13, title: "Her", genre: "Romance", year: 2013, rating: 8.0, watched: false },
];
```

Happy coding!