## Step-by-Step Explanation

### 1. What is the Goal of the Project?

- I created a simple app where users can search for anime movies, pick the ones they like, add them to a list, and manage that list.
- The app fetches data from an external API (Jikan API), displays it dynamically, and lets users interact with it.

---

### 2. How Does the App Work?

#### Step 1: Searching for Anime Movies

- When you type a movie name into the search box and click "Search," the app fetches movie data from the Jikan API.
- It shows the results dynamically in the **Search Results** section.
- If there's no search term or the API fails, the app shows an alert to help users know what's going on.

---

#### Step 2: Displaying Results

- The app has two sections:
    1. **Search Results** (on the left): Shows movies fetched from the API.
    2. **Added Movies** (on the right): A list of movies the user adds.
- These sections are placed side-by-side using Flexbox (a CSS layout tool).
- Each movie is displayed as a "card" with:
    1. A movie poster.
    2. The title.
    3. A link to view more about the movie.

---

#### Step 3: Selecting Movies

- You can click on any movie card to select it.
- Selected movies get a blue border and light background, so it's easy to see which ones are picked.
- Clicking the same card again deselects it (removes the highlight).

---

#### Step 4: Adding Movies to Your List

- Once you select movies from the **Search Results**, you can click the "Add" button.

- The app moves these selected movies to the **Added Movies** section.
- It also ensures no duplicates by checking if a movie is already in the list.

---

#### Step 5: Deleting Movies from Your List

- In the **Added Movies** section, you can select movies and click "Delete."
- The selected movies are removed with a smooth fade-out animation before disappearing.

---

#### Step 6: Resetting the App

- If you want to clear everything and start over, you can click the "Reset" button.
- This will:
    1. Clear all movies from both **Search Results** and **Added Movies**.
    2. Deselect any selected movies.
    3. Reset the search box to be empty.

---

## 3. How Did I Build the App?

#### Step 1: Using the API

- I used the Jikan API to fetch movie data. The API lets me request movie information based on what the user types.
- The app handles errors—if the API doesn't respond or the search box is empty, it shows a message so the user knows what went wrong.

---

#### Step 2: Dynamically Updating the Page

- I used JavaScript to create and update movie cards dynamically:
    - When new data comes in from the API, the app creates cards and displays them in the **Search Results** section.
    - Adding or deleting movies updates the **Added Movies** section in real-time.

---

#### Step 3: Keeping the Code Clean

- I broke the app into small functions, so each one does just one thing:
    - `searchAnime` fetches data from the API.
    - `createMovieCard` builds the movie card HTML.
    - `displaySearchResults` and `displayAddedMovies` show results in the right sections.
    - `resetEverything` clears everything and resets the app.

- This makes the code easy to read, reuse, and debug.

---

## 4. What Did I Learn?

**Key Skills I Demonstrated:**

1. **API Integration**:
    - I learned how to fetch data from an external source and display it dynamically in a user-friendly way.
2. **DOM Manipulation**:
    - I used JavaScript to create, update, and remove elements from the page based on user actions.
3. **Event Handling**:
    - I added interactive features like clicking to select movies, adding/deleting them, and resetting the app.
4. **Error Handling**:
    - The app gracefully handles issues, like an empty search or API errors, by showing helpful alerts.

---

## 5. Why Does This Project Meet the Rubric?

**Proficiency with Learning Goals:**

- The app demonstrates everything required, like API integration, DOM manipulation, and event handling.
- I added extra features like highlighting selections and animations for better usability.

**Technical Communication:**

- The code is clean, uses meaningful names, and has comments explaining how everything works.
- The app gives clear feedback to users with alerts and visual changes.

**Coding Best Practices:**

- The code is modular and reusable, making it easy to maintain or expand.
- I avoided duplicate functionality and handled errors thoughtfully.