

# Practice **Develop SPA Using Angular Components**

# Practice Exercises

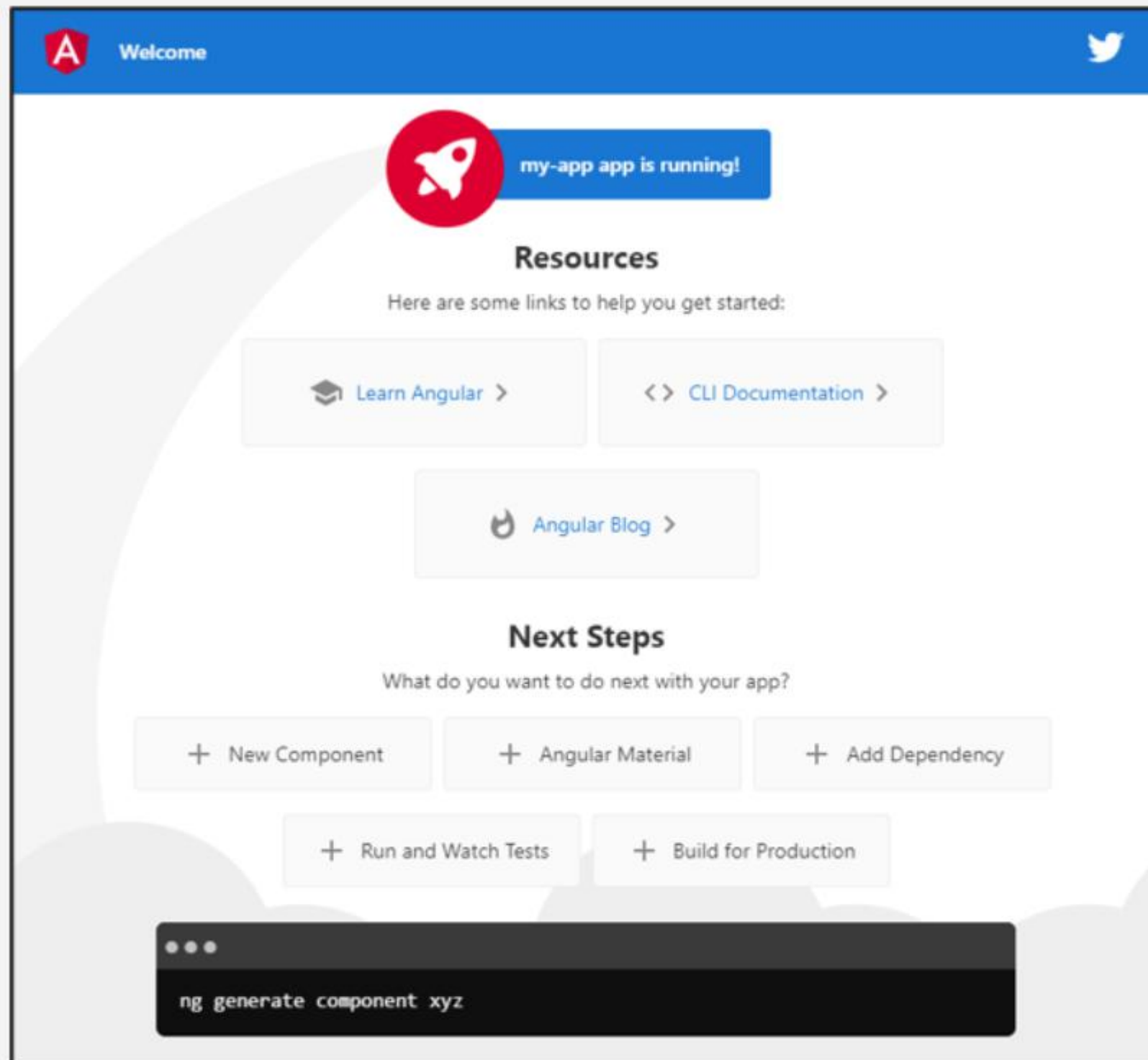
- Practice 1: Create components for Angular Box Office app
- Practice 2: Display a list of trending movies
- Practice 3: Add search movies functionality



# Points to Remember

- Make sure Angular CLI is installed in your local system.
- The resources folder provided with the boilerplate code contains the `movie.ts` and `movies.ts` files required during the development of the solution code.
- Import `FormsModule` in `App.module.ts` to use `ngModel` for implementing 2-way binding.
- Avoid using `any` to declare variables/constants.
- TypeScript code needs to be well indented.





**Angular Application Output created using Angular CLI command**

## Implementation Environment

- Install Angular CLI using the command in the terminal window:
- To create a new workspace and initial starter app, use the "ng new" command, which prompts information about features to include in the initial app.

```
ng new first-app
```

- Run the application with the included server after navigating to the workspace folder.

```
cd first-app  
ng serve -open
```

Note: Delete the default content in the app.component.html to start working with your project.

# Context

Box Office is a popular and authoritative source of movies and TV shows. One can find ratings and reviews for the newest movies and TV shows.

Create a single-page application using Angular CLI that displays an eye-catching list of trending movies to the end users. Movie information like movie name, poster, genre, and ratings should be visible to the user. The users should be able to search for a movie from the list of movies displayed.

# Instructions for the Practice

- [Click here](#) for the boilerplate.
- Please read the README.md file provided in the boilerplate for further instructions about the practice exercise.
- Fork the boilerplate into your own workspace.
- Clone the boilerplate into your local system.
- Open the folder containing the boilerplate code in VS Code.
- Provide the solution code in the files specified with the task details.
- Submit the files (excluding the `node-modules` folder) for manual evaluation.




# Box Office SPA: Expected Output


Box Office

Trending Movies


Go Clear




6.9  
Jurassic World Sci-Fi 2015




7.5  
Doctor Strange Fantasy 2016




6.2  
The Lost City Comedy 2022



8.3  
Spiderman Adventure 2021



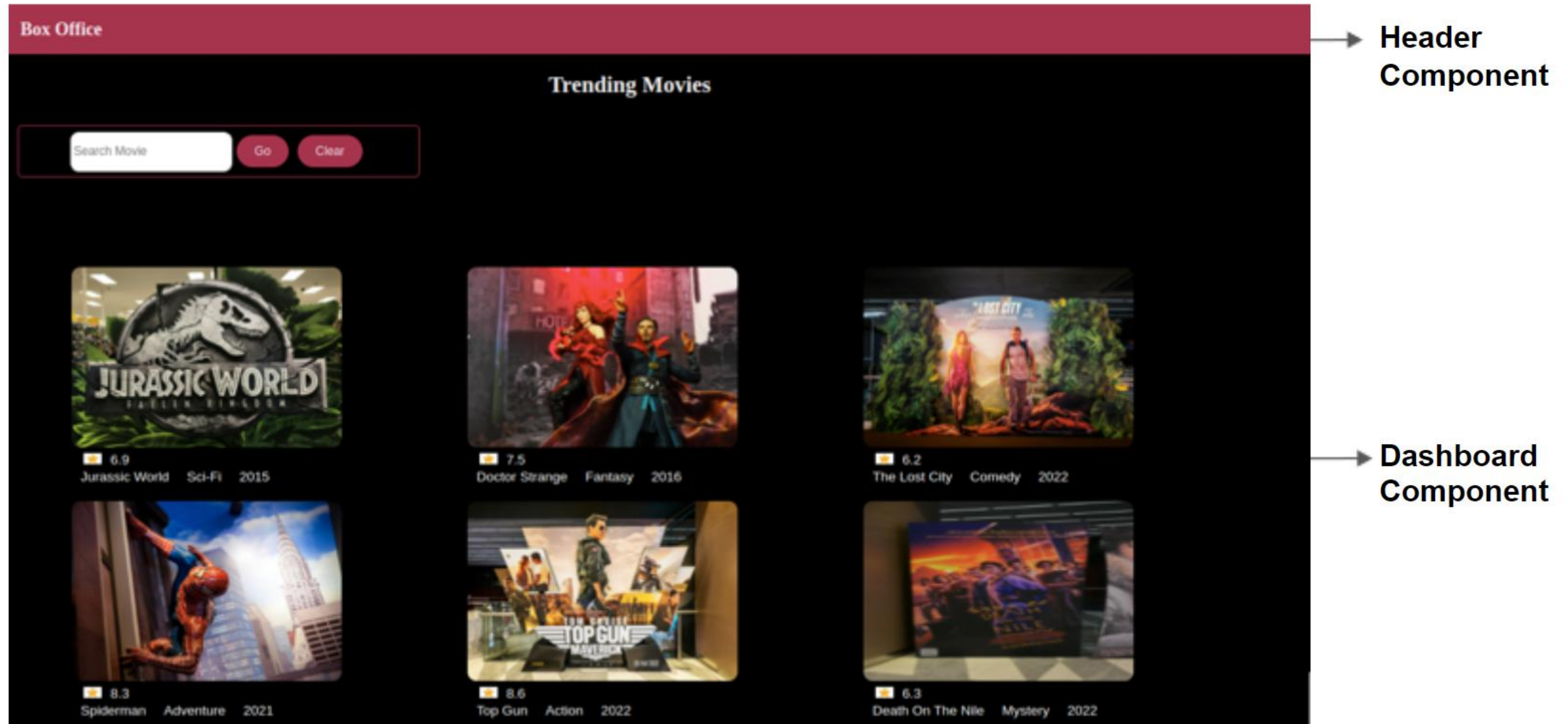
8.6  
Top Gun Action 2022



6.3  
Death On The Nile Mystery 2022

# Identifying Components

- The image below show the components identified for the Box Office App.





An illustration of a woman with dark hair and glasses, wearing a red top, and a man with brown hair and glasses, wearing a yellow top. They are sitting at a desk with a large blue monitor. The woman is holding a yellow clipboard. On the desk, there is a coffee cup, a pencil, and a notepad. The background is light green with some abstract shapes and plants.

## PRACTICE

### **Practice 1: Create Components for Angular Box Office App**

Create an Angular application and components identified inside the application using Angular CLI commands.

Note: Steps to complete this task are given in the upcoming slides.

# Practice 1: Expected Output



# Practice 1: Tasks

- Open the terminal and navigate to the folder '**p1-create-components**'. Create an Angular app named ``Box-Office`` using the Angular CLI command:  

```
`ng new box-office`
```
- Create the identified components for the Box Office app.
- To create a component, use the Angular CLI command.  

```
`ng generate component <component-name>` or `ng g c <component-name>`
```
- Render the component as mentioned below:
  - ``App`` component should render ``Header`` and ``Dashboard`` components
  - ``Header`` component should render the site name ``Box Office``
  - ``Dashboard`` component should display the title "Trending Movies" using text interpolation.
- Design and style the components to generate the output as shown in the ``box-office-output.png`` file.



An illustration of a woman with dark hair and glasses, wearing a red top, and a man with brown hair and glasses, wearing an orange top. They are sitting at a light blue desk with a large blue computer monitor. The woman is holding a yellow clipboard. On the desk, there is a white coffee cup with a red lid, a yellow pencil, and a notepad with a red pencil. The background is light green with some abstract shapes and a large green plant on the right.

## PRACTICE

### Practice 2: Display a List of Trending Movies

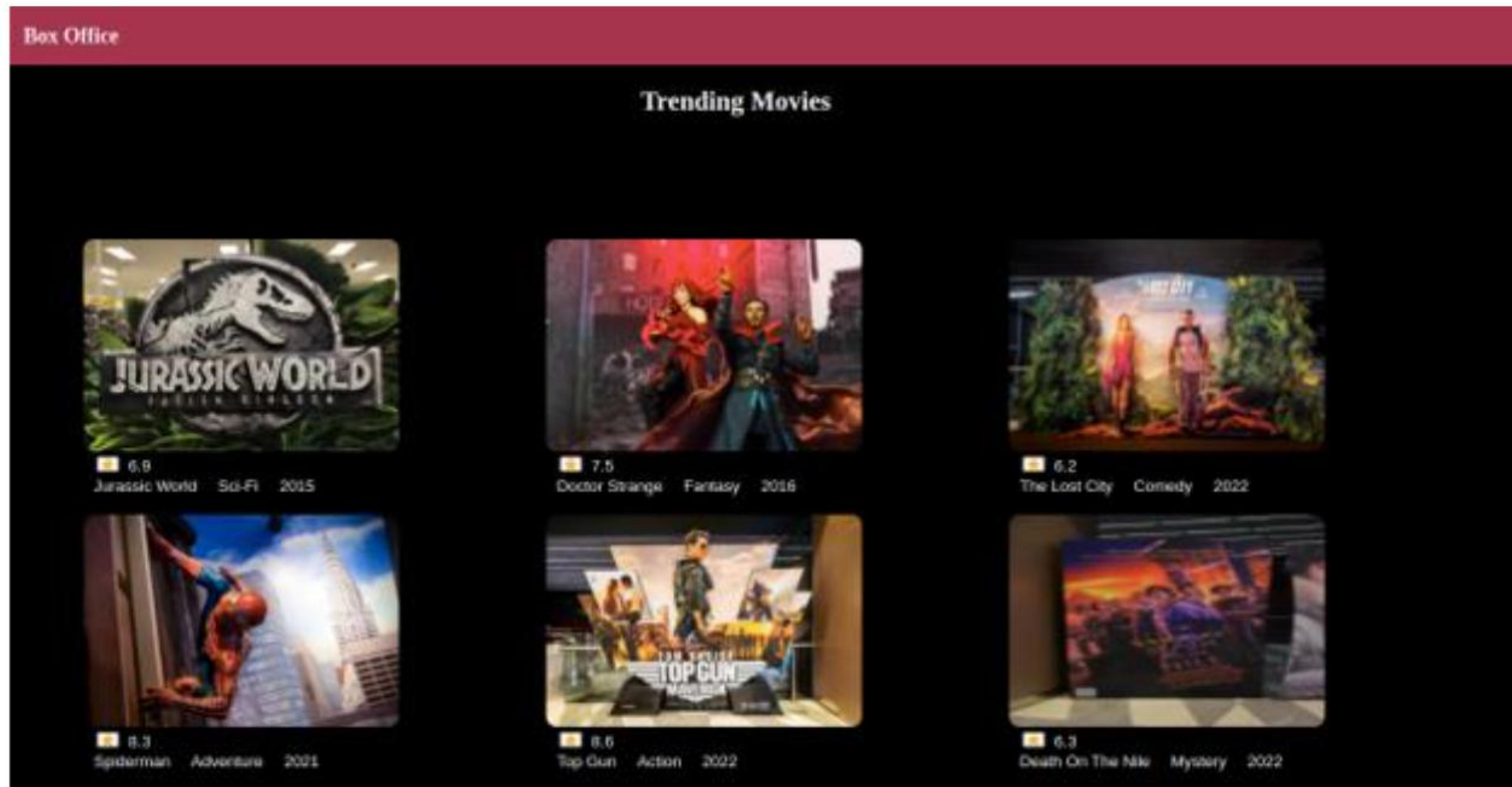
Modify the Box Office app to display the list of trending movies in a tiled format in the dashboard component.

Note: Steps to complete this task are given in the upcoming slides.



# Practice 2: Expected Output

- Copy the images available in the `resources` folder of the boilerplate to `src/assets` folder of the `box-office` app.
- Style the Dashboard component to present the movie list in a grid layout as shown below.



# Practice 2: Tasks

- This practice exercise is in continuation of the previous practice exercise.
- On the command terminal, change the path to folder ``p2-display-movie-list\box-office`` and copy the solution of the ``Box-Office`` app developed in the previous exercise.
- Modify the ``Box-Office`` app code to design a view that displays the trending movies in grid format.
- Create a folder with the name ``models`` under the ``app`` folder of the ``Box-Office`` app.
- Copy the ``movies.ts`` and ``movie.ts`` files from the ``resources`` folder provided with the boilerplate code to the ``models`` folder.
- The ``MOVIES`` array, defined in the ``movies.ts`` file stores the list of objects of type ``Movie.`` The type ``Movie`` is defined in the ``movie.ts`` file.



## Practice 2: Tasks (Cont'd.)

- The dashboard view should be rendered by the `Dashboard` component that reads the movies from the `MOVIES` array and assigns them to the `movies` property of the component class.
- On the `Dashboard` template, iterate through the `movies` property using the ``*ngFor`` directive.
- For each iteration, the `movie` object traversed, the `Dashboard` component should render the `movie` object properties like a movie poster, rating, title, genre, and year of release using Angular text interpolation.
- Design and style the components to generate the output as shown in the `movies-gallery.png` file.

## PRACTICE

### Practice 3: Add Search Movies Functionality

Modify the Box Office app to allow users to search for movies.

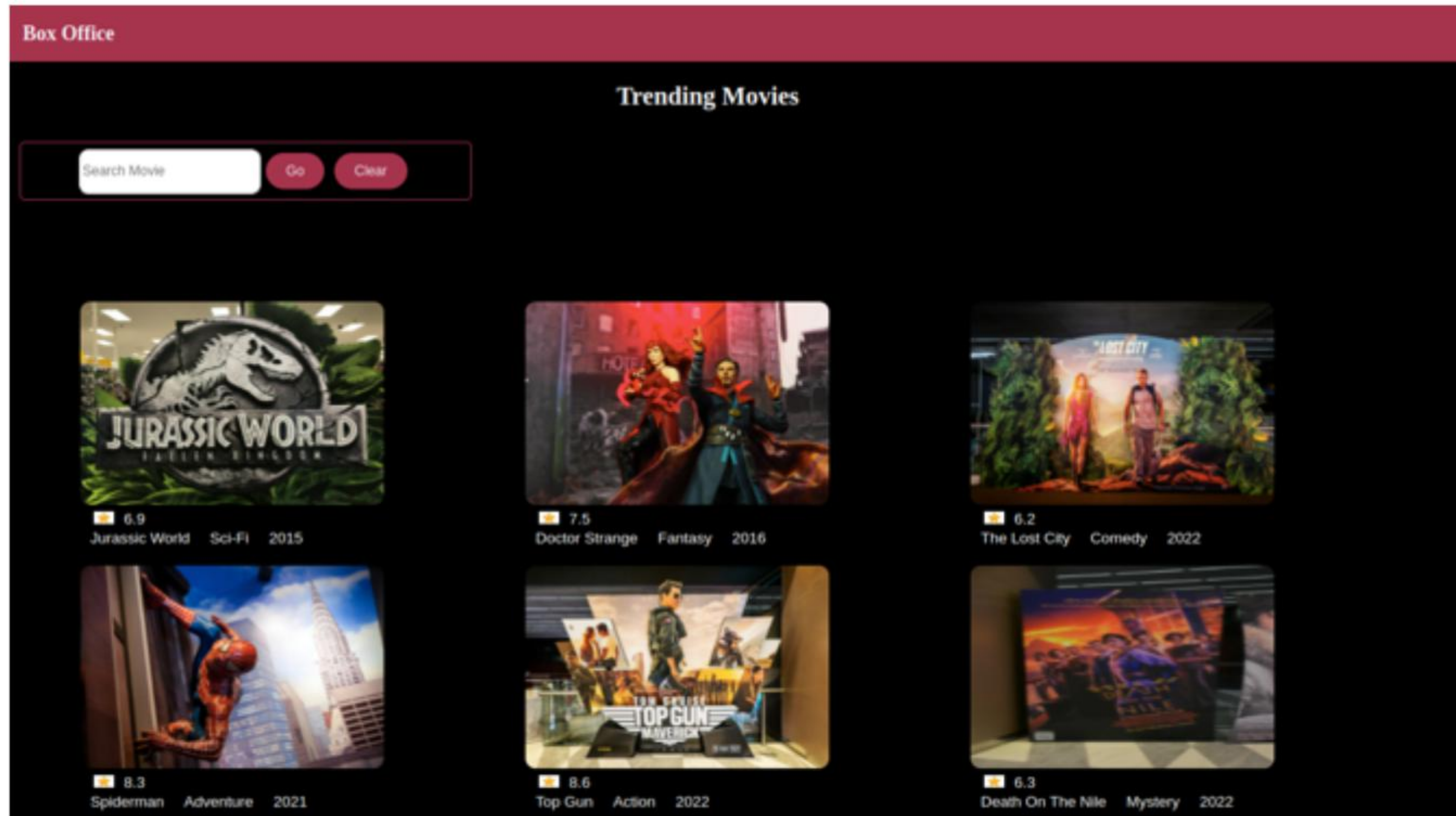
Note: Steps to complete this task are given in the upcoming slides.





# Practice 3: Expected Output

- Below is a snapshot of the expected output:





# Practice 3: Tasks

- This practice exercise is in continuation of the previous practice exercise.
- On the command terminal, change the path to folder ``p3-search-movies\box-office`` and copy the solution of the ``Box-Office`` app developed in the previous exercise.
- Modify the ``Box-Office`` app to add a search feature in the ``Dashboard`` component.
- Declare ``searchText`` as a component property and bind it with the template using `ngModel` to capture the value when the user types in the search text box.
- Add the ``Go`` button, which calls the component's `search()` method whenever the button is clicked using the event binding technique.

```
<button (click) = "search()" type = button>Go</button>
```

- Similarly, add the ``Reset`` button to call the component's `reset()` method to clear any typed text.

```
<button (click) = "reset()" type = button>Reset</button>
```

## Practice 3: Tasks (Cont'd.)

- Write the search logic inside the `search()` method to filter the movies using the Array's filter method.
  - All movies should be displayed when the search text is empty.
- Inside the `reset()` method, set the `searchText` value equal to an empty string and `movies` property value to the `'MOVIES'` array.
- Design and style the components to generate the output as shown in the `'serach-movies.png'` file.