

## React-6 : Imdb-3: TMDB API , watch list and localstorage

### Agenda

Getting Data from TMDB API

Populating movies in Movies Page

Creating the watchlist

Features of Watchlist

Local Storage for keeping data persistent

### title: TMDB API

Replacing dummy entries with data from TMDB

### Analogy

Imagine you're in charge of creating certificates for a course completion. You have a template for the certificate design that includes placeholders for the student's name, course title, and completion date. This template is like React components. React is like your design tool, allowing you to structure and design the certificate layout.

Now, to fill in the actual data and generate the final certificate, you need information about the student who completed the course. This is where APIs come in. APIs act as messengers, fetching the necessary data from a database or external source. In our analogy, the API is

gets the information about the student from various sources and brings it to you.

So, to create the final certificate:

Que We can't use the static images and to drive out our application so we need a server that can give use realtime data.

Solution :

In the above use case we need to talk to servers that just send data in a given format`.

So, to create the final certificate:

**React Components (Template):** You start with React components, just like you start with a certificate template. These components provide the structure and layout for the certificate.

**API Calls (Data Gathering):** Next, you make API calls to gather the specific data needed to fill in the placeholders on the certificate. This is akin to collecting information about the student's name, course details, and completion date from different sources.

**Data Integration (Data into Template):** Once the API returns the necessary data, you integrate it into the React components. This process involves replacing the placeholders in the template with the

actual student data, similar to how you fill in the student's name, course details, and completion date onto the certificate.

**Final UI (Final Certificate):** Finally, with the data integrated into the React components, you have a fully populated certificate – your final UI. This certificate is ready to be presented to the student, showcasing their achievement.

In summary, React provides the framework for designing the certificate layout, while APIs fetch the relevant data to populate the certificate, resulting in a seamless process to generate the final UI, just like creating a personalized certificate for each course completion

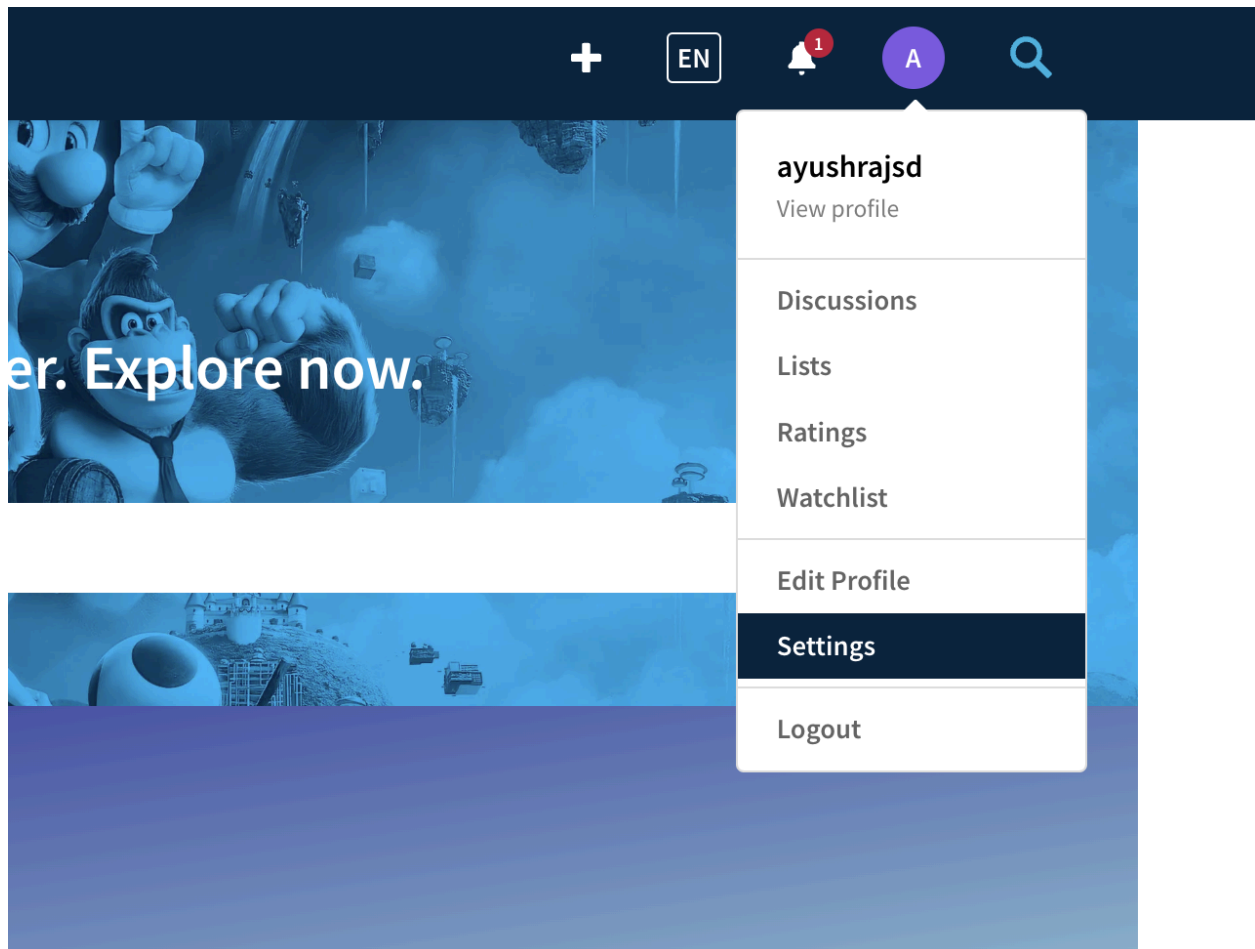
Now ,

To use the TMDB (The Movie Database) API in a React app to get upcoming movies data with images, follow these steps:

Create a TMDB Account and Get an API Key:

1. Go to TMDB and sign up for an account.
2. Navigate to your account settings and find the API section.
3. Create an API key which you will use to make requests to TMDB.

After Signing up go to profile settings

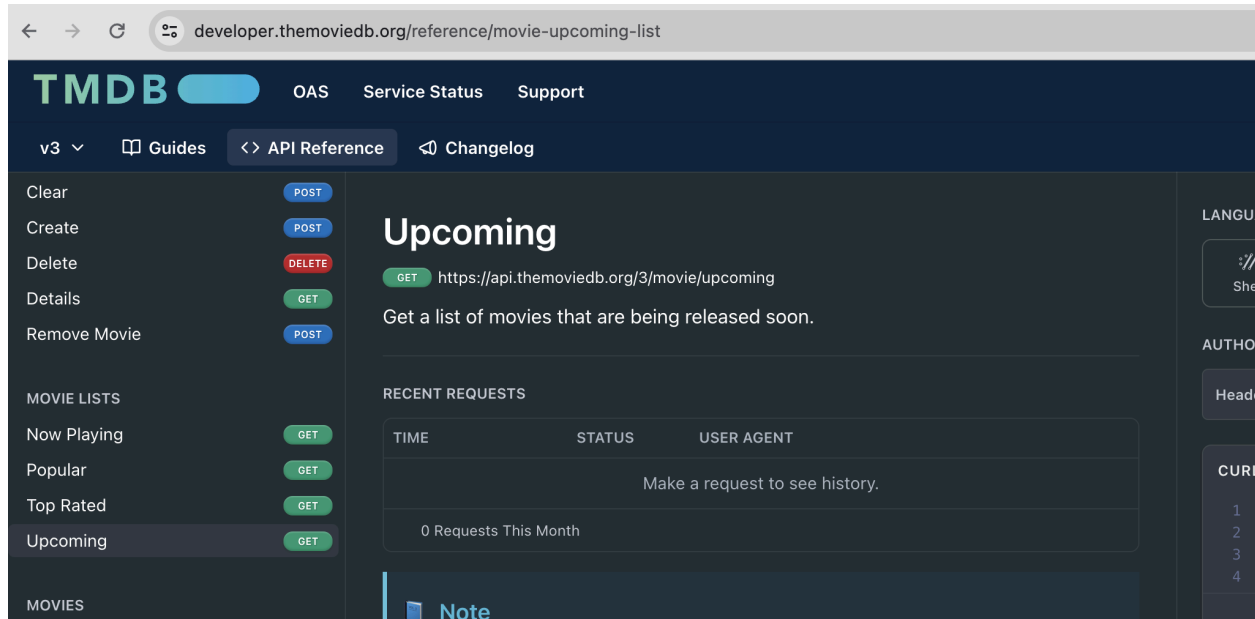


Go to API option and generate a API Key

Now go to API Reference and get started -

<https://developer.themoviedb.org/docs/getting-started>

Explore the reference docs and find relevant links like upcoming movies



## Structure of data received from TMDB API

We will be using this url to get list of movies

[https://api.themoviedb.org/3/trending/movie/day?api\\_key=api\\_key&language=en-US&page=1](https://api.themoviedb.org/3/trending/movie/day?api_key=api_key&language=en-US&page=1)

make sure to add your API key which you generated in the api key field in the url

it will return the data in the form of an array then that will contain lot of properties that populates

```

▼ (20) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}] 1
  ▶ 0: {adult: false, backdrop_path: '/39LxWqvQCsbAe0Cm2B7dtBe3Rd4.jpg', id: 359410, title: 'Road House', original_language: 'en'}
  ▶ 1: {adult: false, backdrop_path: '/xf1rEQRi9pZxoN8HfggVnhj0aBb.jpg', id: 823464, title: 'Godzilla x Kong: The New Empire', original_language: 'en'}
  ▶ 2: {adult: false, backdrop_path: '/TGsfNwKASegCfAn6ED1b08a906.jpg', id: 1125311, title: 'Imaginary', original_language: 'en'}
  ▶ 3: {adult: false, backdrop_path: '/87IVlclAfWL6mdicU1DDuxdwXwe.jpg', id: 693134, title: 'Dune: Part Two', original_language: 'en'}
  ▶ 4: {adult: false, backdrop_path: '/rBo0yYQTTruIYEuXdyCAyVVVlnb2H.jpg', id: 974036, title: 'Ordinary Angels', original_language: 'en'}
  ▶ 5: {adult: false, backdrop_path: '/zAepSr099owYwQqi0QG2AS0dHXw.jpg', id: 634492, title: 'Madame Web', original_language: 'en'}
  ▶ 6: {adult: false, backdrop_path: '/1XDDXPXGiI8id7MrUxK36ke7gkX.jpg', id: 1011985, title: 'Kung Fu Panda 4', original_language: 'en'}
  ▶ 7: {adult: false, backdrop_path: '/3JhQQFidqbrD9k9DxGFCba46EFk.jpg', id: 967847, title: 'Ghostbusters: Frozen Empire', original_language: 'en'}
  ▶ 8: {adult: false, backdrop_path: '/lzWHmYdfeFiMIY4JaMmtR7GELi3.jpg', id: 438631, title: 'Dune', original_language: 'en'}
  ▶ 9: {adult: false, backdrop_path: '/bQS43HSLZzMjZkcHJz4fGc7fNdZ.jpg', id: 792307, title: 'Poor Things', original_language: 'en'}
  ▶ 10: {adult: false, backdrop_path: '/1ZSKH5GGFLM8M32K34GMDaNS2Ew.jpg', id: 802219, title: 'Bob Marley: One Love', original_language: 'en'}
  ▶ 11: {adult: false, backdrop_path: '/deLWk0LZmBNkm8p16igfapQyqeq.jpg', id: 763215, title: 'Damsel', original_language: 'en'}
  ▶ 12: {adult: false, backdrop_path: '/zIYR0rkHJPYB3VTiW1L9QVgaQ0.jpg', id: 897087, title: 'Freelance', original_language: 'en'}
  ▶ 13: {adult: false, backdrop_path: '/nb3xI8XI3w4pMVZ38VijbsyBqP4.jpg', id: 872585, title: 'Oppenheimer', original_language: 'en'}
  ▶ 14: {adult: false, backdrop_path: '/yyFc8Iclt2jxPmLztbP617xXlLT.jpg', id: 787699, title: 'Wonka', original_language: 'en'}
  ▶ 15: {adult: false, backdrop_path: '/keVfqCMKmj55nHzmqR2Q5K7LwJt.jpg', id: 784651, title: 'Fighter', original_language: 'en'}
  ▶ 16: {adult: false, backdrop_path: '/bckxSN9ue0gm0gJpVJmPQrecWuL.jpg', id: 572802, title: 'Aquaman and the Lost Kingdom', original_language: 'en'}
  ▶ 17: {adult: false, backdrop_path: '/fGe1ej335XbqN1j9teoDpofpbLX.jpg', id: 915935, title: 'Anatomy of a Fall', original_language: 'fr'}
  ▶ 18: {adult: false, backdrop_path: '/4dcx4odNyL7IQ8c6vb6oYkjk9rg.jpg', id: 798612, title: 'Shirley', original_language: 'en'}
  ▶ 19: {adult: false, backdrop_path: '/sfKTR7Zi0VZYMLq1waEFeZcNM.jpg', id: 1216512, title: 'The Casagrandes Movie', original_language: 'en'}

```

Here's backdrop\_path property that is responsible for helping you get the poster image

title will give you title of the movie

adult property will tell you if the movie is only for adult or not

In short it is up to you how you want to use this data

## Populating Banner and Movies from TMDB

### Adding dynamic Poster to Banner

1. install axios
2. import axios in Banner component
3. We will next use an useEffect for fetching the data
4. use the fetched data populate our banner component

## Axios Extra notes

Axios is a popular JavaScript library used to make HTTP requests from web browsers or Node.js. It helps you interact with web APIs to fetch or send data. Think of Axios as a tool that helps your web application talk to servers easily.

### Key Features of Axios:

**Easy to Use:** Axios provides a simple syntax for making requests, which makes it easier to write and understand.

**Supports Promises:** Axios uses promises, which allow you to handle asynchronous operations in a cleaner and more readable way.

**Automatic JSON Data Handling:** It automatically transforms JSON data, making it easy to work with APIs that send or receive JSON.

**Built-in Features:** Axios includes features like request and response interception, automatic request retries, and the ability to cancel requests.

## Updating Banner.jsx

Replace placeholder with state variables

```
import React, { useState } from "react";  
function Banner() {  
  const [bannerImage, setBannerImage] = useState("");
```

```

const [title, setTitle] = useState("Placeholder title");

return (
  <div
    className="h-[20vh] md:h-[75vh] bg-cover bg-center flex items-end"
    style={{
      backgroundImage: `url( ${bannerImage} )`,
    }}
  >
    <div className="text-white w-full text-center text-2xl">{title}</div>
  </div>
);
}
export default Banner;

```

Let us add the axios library and the useEffect to make the network call

```

import React, { useEffect, useState } from "react";
import axios from "axios";

function Banner() {
  const [bannerImage, setBannerImage] = useState("");
  const [title, setTitle] = useState("Placeholder title");

  useEffect(() => {
    axios
      .get(
        "https://api.themoviedb.org/3/trending/movie/day?api_key=60d18d673bedf3d701f305ef746f6eef&language=en-US&page=1"
      )
      .then((response) => {
        console.log("Films", response.data.results);
        // for the sake of our example we will be using the first object that
        represents first movie
        const firstMovie = response.data.results[0];
        const firstMovieTitle = firstMovie.title;
        const firstMoviePoster = firstMovie["backdrop_path"];

```



```

        setTitle(firstMovieTitle);

        setBannerImage(
            `https://image.tmdb.org/t/p/original/${firstMoviePoster}`
        );
    });
}, []);

return (
    <div
        className="h-[20vh] md:h-[75vh] bg-cover bg-center flex items-end"
        style={{
            backgroundImage: `url( ${bannerImage} )`,
        }}
    >
        <div className="text-white w-full text-center text-2xl">{title}</div>
    </div>
);
}

export default Banner;

```

## Movies.jsx :

we need to replace the dummy image url and replace that with some realtime url of a movies

1. import axios in movies component
2. We will next use an useEffect for fetching the data
3. set the movies from the incoming data

```

const [movies, setMovies] = useState([]);

useEffect(() => {
    console.log("use effect fetched data");

```

```

    axios
      .get(
        `https://api.themoviedb.org/3/trending/movie/day?api_key=60d18d673bedf3d701f305ef746f6eef&language=en-US&page=1`
      )
      .then(function (res) {
        console.log(res.data.results);
        setMovies(res.data.results);
      });
  }, []);

```

## Updating moviecard.jsx

```

function MovieCard({ movieObj }) {
  return (
    <div
      className="h-[40vh] w-[200px] bg-center bg-cover rounded-xl hover:scale-110
duration-300 hover:cursor-pointer flex flex-col justify-between items-end"
      style={{
        backgroundImage:
`url(https://image.tmdb.org/t/p/original/${movieObj.backdrop_path})`,
      }}
    >
      <div className="text-white w-full text-center text-xl p-2 rounded-lg
bg-gray-900/70">
        {movieObj.title}
      </div>
    </div>
  );
}

export default MovieCard;

```

## Modify the pagination to work with data coming

Let's keep the pagination in sync with movies for that we already have pageNumber state variable .

Now we can use this to update the list of movies as we change the page number.

Solution: Update the useEffect to depend upon pageNo state variable-> add it to dependency array and the url

## Update the useEffect in Movies.jsx page

```
useEffect(() => {
  console.log("use effect fetched data");
  axios
    .get(
      `https://api.themoviedb.org/3/trending/movie/day?api_key=60d18d673bedf3d701f305ef746f6eef&language=en-US&page=${pageNo}`
    )
    .then(function (res) {
      console.log(res.data.results);
      setMovies(res.data.results);
    });
}, [pageNo]);
```

title: Add to watch list feature

Add/remove movies from watch List

Add to Watch list : by default no movies will be added to watch list and 🥰 is visible on top right and as you click on that it should add the movie to watchlist and update it with cross



remove movie from watch List: exact opposite of the above feature should happen

### Add to watch list

We will start by initializing this as an empty array. One by one, we will add the movie object , and then we will retrieve them to show list of added movies

```
function Movies() {  
  const [watchList, setWatchList] = useState([]);
```

One approach is to use filtering based on the movie's ID. To begin, we will add a button to our movie card. Inside the card div, we will create another div to accommodate this button.

In the movie card, add the div for emoji ( ctrl + cmd + space )

```
<div className="">🥰</div>  
  <div className="text-white w-full text-center text-xl p-2 rounded-lg  
bg-gray-900/70">
```

```
{movieObj.title}
```

Let us style this a bit

```
<div className="m-4 flex justify-center h-8 w-8 items-center  
rounded-lg bg-gray-900/60">🤪</div>
```

In CSS, the Tailwind properties would translate roughly as follows:

1. m-4 applies a margin of 1rem (since the default Tailwind scale is 0.25rem per unit) to all sides.
2. flex sets the display to flex, enabling flexbox layout.
3. justify-center centers the flex items along the main axis (horizontally).

4. items-center centers the flex items along the cross axis (vertically).
5. h-8 and w-8 set both the height and width to 2rem (8 units \* 0.25rem).
6. rounded-lg applies a moderate border radius, making the edges rounded.
7. bg-gray-900/60 sets the background color to a very dark gray with 60% opacity.

If a movie has not been added to the watchlist, you should display an emoji. If it's already on the watchlist, you should display a cross sign.

As of now, the watchlist is empty.

The purpose of adding a movie to the watchlist is that when we click on the emoji, a function should be triggered to provide us the movie we've clicked on.

```
const addToWatchList = (movieObj) => {  
  const updatedWatchlist = [...watchList, movieObj];  
  setWatchList(updatedWatchlist);  
};
```

Pass this as props to Movie card

```
return <MovieCard movieObj={movieObj} addToWatchList={addToWatchList}/>;
```

Update MovieCard

```
function MovieCard({ movieObj, addToWatchList }) {
```

```

<div
  onClick={() => addToWatchList(movieObj)}
  className="m-4 flex justify-center h-8 w-8 items-center rounded-lg
bg-gray-900/60"
  >
    🚫
  </div>

```

## Remove from watchlist

Next, we will examine the functionality of a cross button for removing items from the watchlist, involving conditional code.

From the Movies.jsx pass the watchList as props

```

return <MovieCard movieObj={movieObj} addToWatchList={addToWatchList}
watchList={watchList}/>;

```

Now in the movie card we implement a function that checks if the given movie object exists in the watch list or not

```

function MovieCard({ movieObj, addToWatchList, watchList }) {
  const doesContain = (movieObject) => {
    for (let i = 0; i < watchList.length; i++) {
      if (watchList[i].id === movieObject.id) {
        return true; // chnage button to cross
      }
    }
    return false; // added to my WatchList
  }
  return (
    <div
      className="h-[40vh] w-[200px] bg-center bg-cover rounded-xl hover:scale-110
duration-300 hover:cursor-pointer flex flex-col justify-between items-end"
      style={{

```

```

      backgroundImage:
`url (https://image.tmdb.org/t/p/original/${movieObj.backdrop_path})`,
    }}
  >
  {
    doesContain(movieObj) ? (
      <div
        onClick={() => addToWatchList(movieObj)}
        className="m-4 flex justify-center h-8 w-8 items-center rounded-lg
bg-red-900/60"
      >
        ✖
      </div>
    ) : (
      <div
        onClick={() => addToWatchList(movieObj)}
        className="m-4 flex justify-center h-8 w-8 items-center rounded-lg
bg-gray-900/60"
      >
        🤖
      </div>
    )
  }

  <div className="text-white w-full text-center text-xl p-2 rounded-lg
bg-gray-900/70">
    {movieObj.title}
  </div>
</div>
);
}

export default MovieCard;

```

What we have done is make the icon as conditional



Now, we can proceed to create the "Remove from Watchlist" function.


In the movies.jsx, add the code below

```
const removeFromWatchList = (movieObj) => {  
  let filteredMovies = watchList.filter((movie) => {  
    return movie.id !== movieObj.id;  
  });  
  setWatchList(filteredMovies);  
};
```

Pass it as props

```
<MovieCard  
  movieObj={movieObj}  
  addToWatchList={addToWatchList}  
  watchList={watchList}  
  removeFromWatchList={removeFromWatchList}  
>
```

```
function MovieCard({ movieObj, addToWatchList, watchList, removeFromWatchList }) {
```

```
<div  
  onClick={() => removeFromWatchList(movieObj)}  
  className="m-4 flex justify-center h-8 w-8 items-center rounded-lg  
bg-red-900/60"  
  >  
      
  </div>
```

