Comprehensive Class Notes: Local Storage and TMDB API Integration

Welcome to the class on Local Storage and TMDB API Integration, where we'll explore how to use local storage in web applications and integrate a movie database API to fetch and display movie data. This class is essential for both understanding persistent client-side storage and for handling data dynamically using APIs.

1. Introduction to Local Storage

Definition: Local Storage is a web storage feature that allows the persistence of data in a browser, which means data will remain stored even after the browser is closed and reopened.

Key Characteristics:

- **Persistent Storage:** Data remains after the browser is closed.
- **Capacity Limit:** Typically limited to about 5MB per origin, although this can vary slightly between browsers.
- **Cross-page Accessibility:** Accessible across all pages within the same domain.
- **No Expiry:** Data stored in local storage has no expiry date and will remain there until explicitly deleted by JavaScript or until the user clears it through their browser settings.

CRUD Operations in Local Storage

- **Create:** `localStorage.setItem('key', 'value')` - Stores a key-value pair.
- **Read:** `let value = localStorage.getItem('key')` - Retrieves the value associated with the key.

associated with the key.

- **Clear All:** `localStorage.clear()` - Clears all items stored 【8:0†transcript.txt】.

JSON Serialization

Local Storage can only store strings. Therefore, complex data types (e.g., objects or arrays) must be converted to a JSON string using `JSON.stringify()` before storage. Conversely, `JSON.parse()` is used to convert the string back to its original format when retrieving 【8:5†transcript.txt】.

2. Practical Application of Local Storage

In the class, we created a watchlist feature for a movie application:

- **Adding Movies:** Movies can be added to a watchlist by storing them in local storage.
- **Retrieving Watchlist:** On page load, retrieve movies from local storage to populate the watchlist in the UI 【8:18†transcript.txt】.

Example Code for Adding and Retrieving Watchlist

```
function addToWatchList(movieObj) {
    let updatedWatchlist = [...watchList, movieObj];
    setWatchList(updatedWatchlist);
    localStorage.setItem('movies', JSON.stringify(updatedWat
}

useEffect(() => {
    let moviesFromLocalStorage = localStorage.getItem('movie
    if (moviesFromLocalStorage) {
        setWatchList(JSON.parse(moviesFromLocalStorage));
```

Scaler Companion beta

3. Integrating TMDB API

APIs and Their Importance: APIs (Application Programming Interfaces) allow different software applications to communicate. APIs like TMDB provide access to extensive databases which can be queried for dynamic data【8:3†transcript.txt】【8:6†transcript.txt】.

Using Axios for API Calls:

- **Axios** is a promise-based HTTP client for making HTTP requests. It's widely used for its simplicity and ease of error handling.

Example of Fetching Movie Data with Axios

```
import axios from 'axios';

useEffect(() => {
    axios.get('https://api.themoviedb.org/3/trending/movie/d
        params: { api_key: 'YOUR_API_KEY' }
    })
    .then(response => {
        setMovies(response.data.results);
    })
    .catch(error => {
        console.error("There was an error fetching the movie
    });
}, []);
```

Handling Pagination with API: We implemented pagination by modifying the API URL to include a page parameter, so different sets of movies can be fetched as the user navigates through pages【8:7†transcript.txt】【8:17†transcript.txt】.

In this class, we learned how to leverage local storage to maintain data across browser sessions, which is crucial for creating seamless user experiences. We also explored how to fetch and display dynamic data using APIs, which is key to building interactive and data-driven web applications. This integrated approach enables the building of robust applications capable of persisting user data and dynamically fetching real-time content.

Homework/Practice: Try building a small web application that allows users to bookmark their favorite articles, utilizing both local storage for persistence and an external API for fetching article data.