EXPERIMENT NO - 03

- **AIM:**

Manage complex state with Redux or Context API

- **THEORY:**

1. Redux vs Context API Deep Dive

Redux: Predictable state container with single source of truth, immutable updates, and pure reducer functions
Context API: React's built-in solution for prop drilling elimination with provider/consumer pattern
Detailed comparison table showing when to use each approach

2. Implementation in This Experiment

Why Context API was chosen:
Medium complexity application with clear domain boundaries
Faster development without Redux boilerplate
Demonstrates React's built-in capabilities
Good performance with proper optimization

Multi-Context Architecture:
ShowsContext: Manages show catalog, filtering, sorting, search
UserContext: Handles profiles, preferences, watch history, ratings

3. Advanced Patterns Demonstrated

Core Patterns:

Reducer Pattern: Complex state logic with useReducer
Custom Hooks: Business logic abstraction with useShowActions
Cross-Context Communication: Seamless data flow between contexts
Optimistic Updates: Immediate UI feedback

Extra Features:

Continue Watching: Smart progress tracking with visual indicators
User Statistics Dashboard: Real-time metrics and analytics
Advanced Multi-Criteria Filtering: Genre, year, rating, type combinations
Interactive Rating System: Modal-based ratings with persistence
Theme Management: Persistent dark/light mode with smooth transitions
Micro-Interactions: Hover effects, animations, contextual UI states

4. Performance Optimizations

Context separation to prevent unnecessary re-renders
useReducer for batched state updates
Custom hooks for business logic encapsulation
Potential improvements with memoization and virtualization

5. Interactive Theory Demo
Added a new /theory route with tabs showing:

Architecture: Live state metrics and Context vs Redux comparison
Patterns: Code examples of advanced patterns implemented
Creativity: Showcase of the 30% extra features
Performance: Optimization strategies and scalability considerations
The implementation demonstrates how Context API can handle sophisticated state management requirements while maintaining code clarity and performance, making it suitable for real-world applications like content management systems, e-commerce platforms, and social media applications.

● **SOURCE CODE:**

```tsx
import React from 'react';
import { Play, Clock } from 'lucide-react';
import { useUser } from '../contexts/UserContext';
import { useShows } from '../contexts/ShowsContext';
import { useShowActions } from '../hooks/useShowActions';

const ContinueWatching: React.FC = () => {
  const { state: userState } = useUser();
  const { state: showsState } = useShows();
  const { updateWatchProgress } = useShowActions();

  const continueWatchingShows = userState.watchHistory
    .filter(item => item.progress > 0 && item.progress < 100)
    .sort((a, b) => new Date(b.lastWatched).getTime() - new Date(a.lastWatched).getTime())
    .slice(0, 4)
    .map(watchItem => {
      const show = showsState.shows.find(s => s.id === watchItem.showId);
      return show ? { ...show, watchProgress: watchItem } : null;
    })
    .filter(Boolean);

  if (continueWatchingShows.length === 0) {
    return null;
  }
```

FilterPanel.tsx 2 | ContinueWatching.tsx 1 | App.tsx | index.ts | useShowActions.ts 1 | index.css 3 | main

src > components > FilterPanel.tsx > FilterPanel

```tsx
import React, { useState } from 'react';
import { Filter, ChevronDown, ChevronUp } from 'lucide-react';
import { useShows } from '../contexts/ShowsContext';
import { useUser } from '../contexts/UserContext';
import { useShowActions } from '../hooks/useShowActions';
import { genres, maturityRatings } from '../data/mockData';

const FilterPanel: React.FC = () => {
  const { state: showsState } = useShows();
  const { state: userState } = useUser();
  const { filterShows, sortShows } = useShowActions();
  const [isExpanded, setIsExpanded] = useState(false);

  const handleFilterChange = (filterType: keyof typeof showsState.filters, value: any) => {
    const newFilters = {
      ...showsState.filters,
      [filterType]: value
    };
    filterShows(newFilters);
  };

  const handleGenreToggle = (genre: string) => {
    const currentGenres = showsState.filters.genre;
    const newGenres = currentGenres.includes(genre)
      ? currentGenres.filter(g => g !== genre)
      : [...currentGenres, genre];
```

```tsx
import React, { createContext, useContext, useReducer, ReactNode } from 'react';

import { UserProfile, WatchProgress, UserRating } from '../types';



interface UserState {

  currentProfile: UserProfile | null;

  profiles: UserProfile[];

  watchHistory: WatchProgress[];

  userRatings: UserRating[];

  theme: 'light' | 'dark';

}
```

```ts
type UserAction =

  | { type: 'SET_CURRENT_PROFILE'; payload: UserProfile }

  | { type: 'ADD_PROFILE'; payload: UserProfile }

  | { type: 'UPDATE_PROFILE'; payload: UserProfile }

  | { type: 'DELETE_PROFILE'; payload: string }

  | { type: 'ADD_TO_WATCHLIST'; payload: string }

  | { type: 'REMOVE_FROM_WATCHLIST'; payload: string }

  | { type: 'ADD_TO_FAVORITES'; payload: string }

  | { type: 'REMOVE_FROM_FAVORITES'; payload: string }

  | { type: 'UPDATE_WATCH_PROGRESS'; payload: WatchProgress }

  | { type: 'ADD_USER_RATING'; payload: UserRating }

  | { type: 'TOGGLE_THEME' };



const initialProfiles: UserProfile[] = [

  {

    id: '1',

    name: 'John Doe',

    avatar: '□',

    preferences: {

      genres: ['Sci-Fi', 'Thriller'],

      maturityLevel: 'TV-MA',

      autoplay: true,
```

```
      subtitles: false
    },
    watchlist: ['1', '3', '5'],
    favorites: ['1', '2']
  },
  {
    id: '2',
    name: 'Jane Smith',
    avatar: '□',
    preferences: {
      genres: ['Drama', 'Romance'],
      maturityLevel: 'PG-13',
      autoplay: false,
      subtitles: true
    },
    watchlist: ['2', '7', '10'],
    favorites: ['7', '10']
  }
];

const initialState: UserState = {
  currentProfile: initialProfiles[0],
```

```
  profiles: initialProfiles,

  watchHistory: [

    { showId: '1', progress: 75, lastWatched: new Date(), currentSeason:
1, currentEpisode: 6 },

    { showId: '2', progress: 30, lastWatched: new Date(Date.now() -
86400000) },

    { showId: '3', progress: 90, lastWatched: new Date(Date.now() -
172800000) }

  ],

  userRatings: [

    { showId: '1', rating: 9, review: 'Amazing show!' },

    { showId: '2', rating: 8 },

    { showId: '3', rating: 10, review: 'Mind-blowing series' }

  ],

  theme: 'dark'

};



const userReducer = (state: UserState, action: UserAction): UserState =>
{

  switch (action.type) {

    case 'SET_CURRENT_PROFILE':

      return { ...state, currentProfile: action.payload };

    case 'ADD_PROFILE':

      return { ...state, profiles: [...state.profiles, action.payload]
```

```
};

    case 'UPDATE_PROFILE': {

      const updatedProfiles = state.profiles.map(profile =>

        profile.id === action.payload.id ? action.payload : profile

      );

      return {

        ...state,

        profiles: updatedProfiles,

        currentProfile: state.currentProfile?.id === action.payload.id ?
action.payload : state.currentProfile

      };

    }

    case 'DELETE_PROFILE':

      return {

        ...state,

        profiles:  state.profiles.filter(profile  =>  profile.id  !==
action.payload),

        currentProfile:  state.currentProfile?.id  ===  action.payload  ?
state.profiles[0] : state.currentProfile

      };

    case 'ADD_TO_WATCHLIST': {

      if (!state.currentProfile) return state;

      const updatedProfile = {
```

```
        ...state.currentProfile,
        watchlist: [...state.currentProfile.watchlist, action.payload]
      };
      return userReducer(state, { type: 'UPDATE_PROFILE', payload: updatedProfile });
    }
    case 'REMOVE_FROM_WATCHLIST': {
      if (!state.currentProfile) return state;
      const updatedProfile = {
        ...state.currentProfile,
        watchlist: state.currentProfile.watchlist.filter(id => id !== action.payload)
      };
      return userReducer(state, { type: 'UPDATE_PROFILE', payload: updatedProfile });
    }
    case 'ADD_TO_FAVORITES': {
      if (!state.currentProfile) return state;
      const updatedProfile = {
        ...state.currentProfile,
        favorites: [...state.currentProfile.favorites, action.payload]
      };
      return userReducer(state, { type: 'UPDATE_PROFILE', payload: updatedProfile });
```

```
    }

    case 'REMOVE_FROM_FAVORITES': {

      if (!state.currentProfile) return state;

      const updatedProfile = {

        ...state.currentProfile,

        favorites: state.currentProfile.favorites.filter(id => id !==
action.payload)

      };

      return userReducer(state, { type: 'UPDATE_PROFILE', payload:
updatedProfile });

    }

    case 'UPDATE_WATCH_PROGRESS': {

      const existingIndex = state.watchHistory.findIndex(item =>
item.showId === action.payload.showId);

      let newWatchHistory;



      if (existingIndex >= 0) {

        newWatchHistory = [...state.watchHistory];

        newWatchHistory[existingIndex] = action.payload;

      } else {

        newWatchHistory = [...state.watchHistory, action.payload];

      }
```

```
      return { ...state, watchHistory: newWatchHistory };

    }

    case 'ADD_USER_RATING': {

      const existingIndex = state.userRatings.findIndex(rating =>
rating.showId === action.payload.showId);

      let newRatings;



      if (existingIndex >= 0) {

        newRatings = [...state.userRatings];

        newRatings[existingIndex] = action.payload;

      } else {

        newRatings = [...state.userRatings, action.payload];

      }



      return { ...state, userRatings: newRatings };

    }

    case 'TOGGLE_THEME':

      return { ...state, theme: state.theme === 'dark' ? 'light' : 'dark'
};

    default:

      return state;

  }

};
```

```tsx
const UserContext = createContext<{

  state: UserState;

  dispatch: React.Dispatch<UserAction>;

} | null>(null);



export const UserProvider: React.FC<{ children: ReactNode }> = ({ children
}) => {

  const [state, dispatch] = useReducer(userReducer, initialState);



  return (

    <UserContext.Provider value={{ state, dispatch }}>

      {children}

    </UserContext.Provider>

  );

};



export const useUser = () => {

  const context = useContext(UserContext);

  if (!context) {

    throw new Error('useUser must be used within a UserProvider');

  }
```

```
  return context;

};

import { Show } from '../types';



export const mockShows: Show[] = [

  {

    id: '1',

    title: 'Stranger Things',

    genre: 'Sci-Fi',

    year: 2016,

    rating: 8.7,

    duration: '51min',

    description: 'When a young boy vanishes, a small town uncovers a mystery involving secret experiments.',

    imageUrl:    'https://images.pexels.com/photos/7991579/pexels-photo-7991579.jpeg',

    seasons: 4,

    episodes: 34,

    type: 'series',

    maturityRating: 'TV-MA',

    tags: ['supernatural', 'thriller', 'nostalgia', 'friendship']

  },

  {
```

```
    id: '2',
    title: 'The Crown',
    genre: 'Drama',
    year: 2016,
    rating: 8.6,
    duration: '58min',
    description: 'Follows the political rivalries and romance of Queen Elizabeth IIs reign.',
    imageUrl:    'https://images.pexels.com/photos/8111367/pexels-photo-8111367.jpeg',
    seasons: 6,
    episodes: 60,
    type: 'series',
    maturityRating: 'TV-MA',
    tags: ['historical', 'royal', 'political', 'biographical']
  },
  {
    id: '3',
    title: 'Black Mirror',
    genre: 'Sci-Fi',
    year: 2011,
    rating: 8.8,
    duration: '45min',
```

```
    description: 'An anthology series exploring the dark aspects of modern technology.',

    imageUrl:    'https://images.pexels.com/photos/3861969/pexels-photo-3861969.jpeg',

    seasons: 6,

    episodes: 27,

    type: 'series',

    maturityRating: 'TV-MA',

    tags: ['dystopian', 'technology', 'psychological', 'anthology']

  },

  {

    id: '4',

    title: 'The Witcher',

    genre: 'Fantasy',

    year: 2019,

    rating: 8.2,

    duration: '60min',

    description: 'Geralt of Rivia, a mutated monster-hunter for hire, journeys toward his destiny.',

    imageUrl:    'https://images.pexels.com/photos/4031818/pexels-photo-4031818.jpeg',

    seasons: 3,

    episodes: 24,

    type: 'series',
```

```
    maturityRating: 'TV-MA',
    tags: ['fantasy', 'adventure', 'magic', 'medieval']
  },
  {
    id: '5',
    title: 'Squid Game',
    genre: 'Thriller',
    year: 2021,
    rating: 8.0,
    duration: '56min',
    description: 'Hundreds of cash-strapped players accept an invitation to compete in deadly games.',
    imageUrl:    'https://images.pexels.com/photos/7991225/pexels-photo-7991225.jpeg',
    seasons: 2,
    episodes: 17,
    type: 'series',
    maturityRating: 'TV-MA',
    tags: ['survival', 'psychological', 'korean', 'social-commentary']
  },
  {
    id: '6',
    title: 'Money Heist',
```

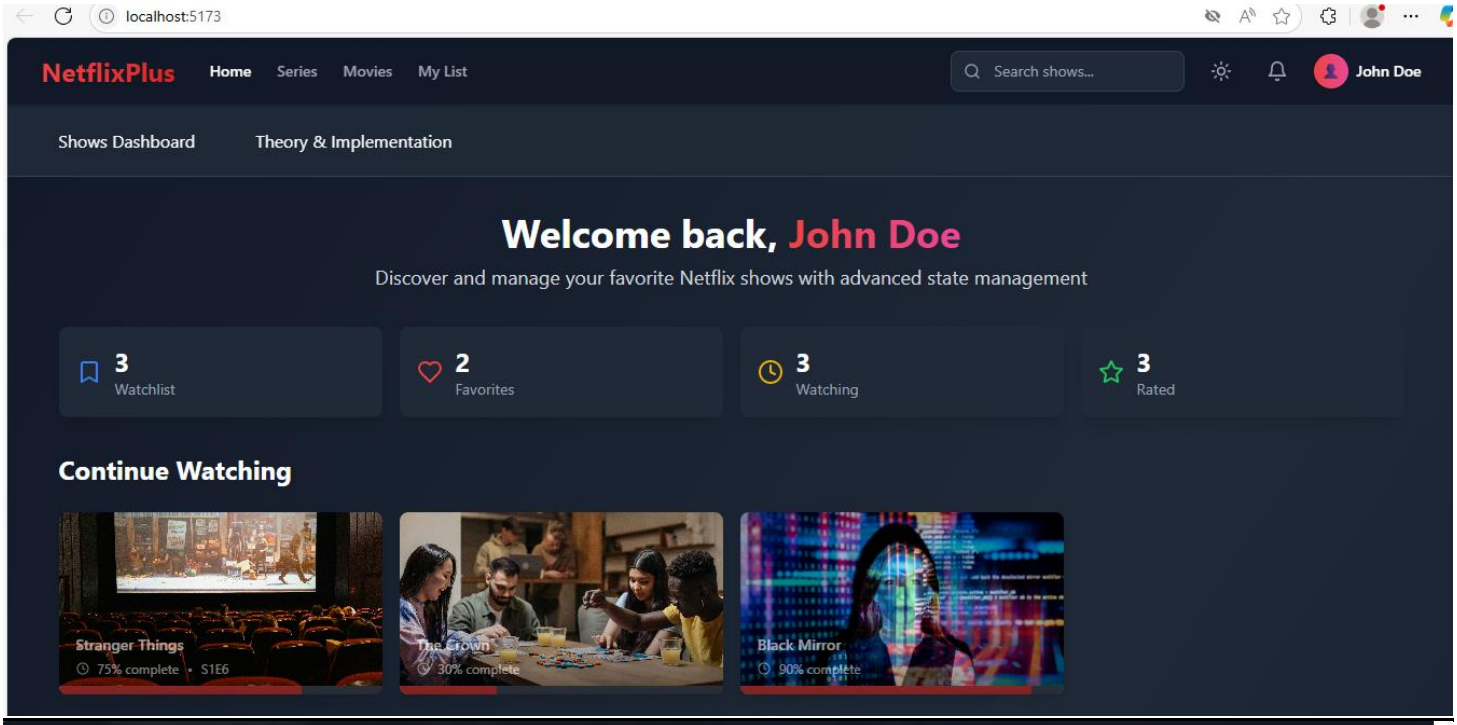
```
    genre: 'Crime',

    year: 2017,

    rating: 8.3,

    duration: '70min',

    description: 'An unusual group of robbers attempt to carry out the most perfect robbery in Spanish history.',

    imageUrl:    'https://images.pexels.com/photos/8111226/pexels-photo-8111226.jpeg',

    seasons: 5,

    episodes: 41,

    type: 'series',

    maturityRating: 'TV-MA',

    tags: ['heist', 'spanish', 'strategy', 'drama']

  },

  {

    id: '7',

    title: 'Bridgerton',

    genre: 'Romance',

    year: 2020,

    rating: 7.3,

    duration: '62min',

    description: 'The romantic and scandalous world of the Bridgerton family in Regency-era England.',
```

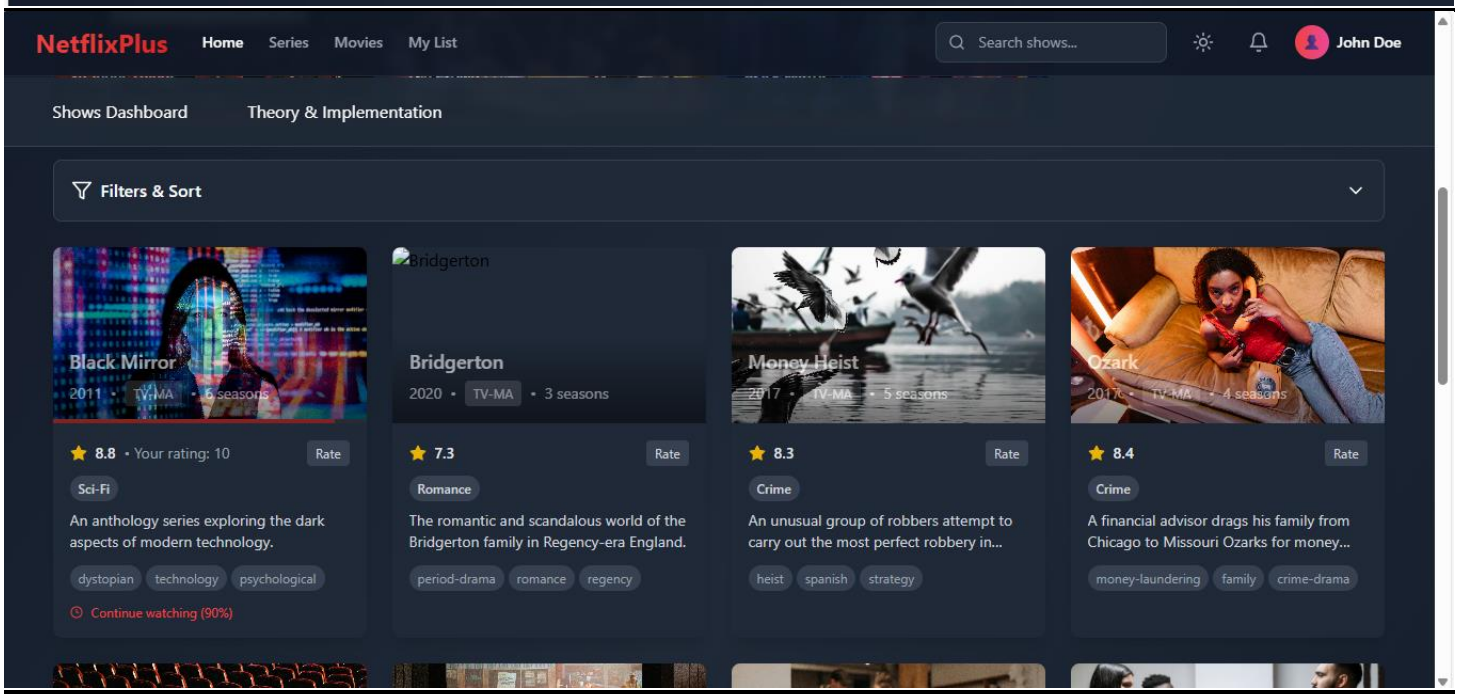
```
    imageUrl:    'https://images.pexels.com/photos/8111193/pexels-photo-8111193.jpeg',

    seasons: 3,

    episodes: 24,

    type: 'series',

    maturityRating: 'TV-MA',

    tags: ['period-drama', 'romance', 'regency', 'family']

  },

  {

    id: '8',

    title: 'Wednesday',

    genre: 'Comedy',

    year: 2022,

    rating: 8.1,

    duration: '45min',

    description: 'Wednesday Addams navigates her years as a student at Nevermore Academy.',

    imageUrl:    'https://images.pexels.com/photos/7991498/pexels-photo-7991498.jpeg',

    seasons: 1,

    episodes: 8,

    type: 'series',

    maturityRating: 'PG-13',
```

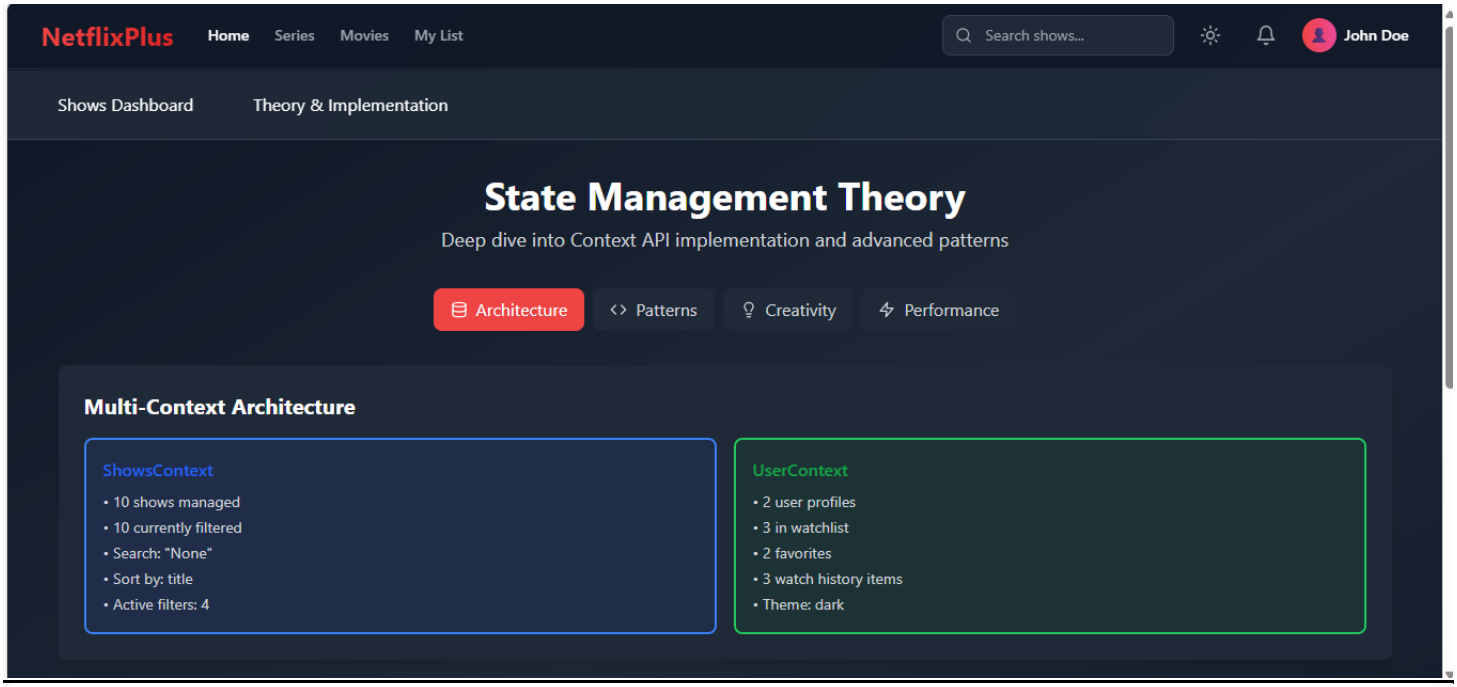
```
    tags: ['supernatural', 'teen', 'mystery', 'comedy']

  },

  {

    id: '9',

    title: 'Ozark',

    genre: 'Crime',

    year: 2017,

    rating: 8.4,

    duration: '60min',

    description: 'A financial advisor drags his family from Chicago to
Missouri Ozarks for money laundering.',

    imageUrl:    'https://images.pexels.com/photos/8111342/pexels-photo-
8111342.jpeg',

    seasons: 4,

    episodes: 44,

    type: 'series',

    maturityRating: 'TV-MA',

    tags: ['money-laundering', 'family', 'crime-drama', 'dark']

  },

  {

    id: '10',

    title: 'The Queen\'s Gambit',

    genre: 'Drama',
```

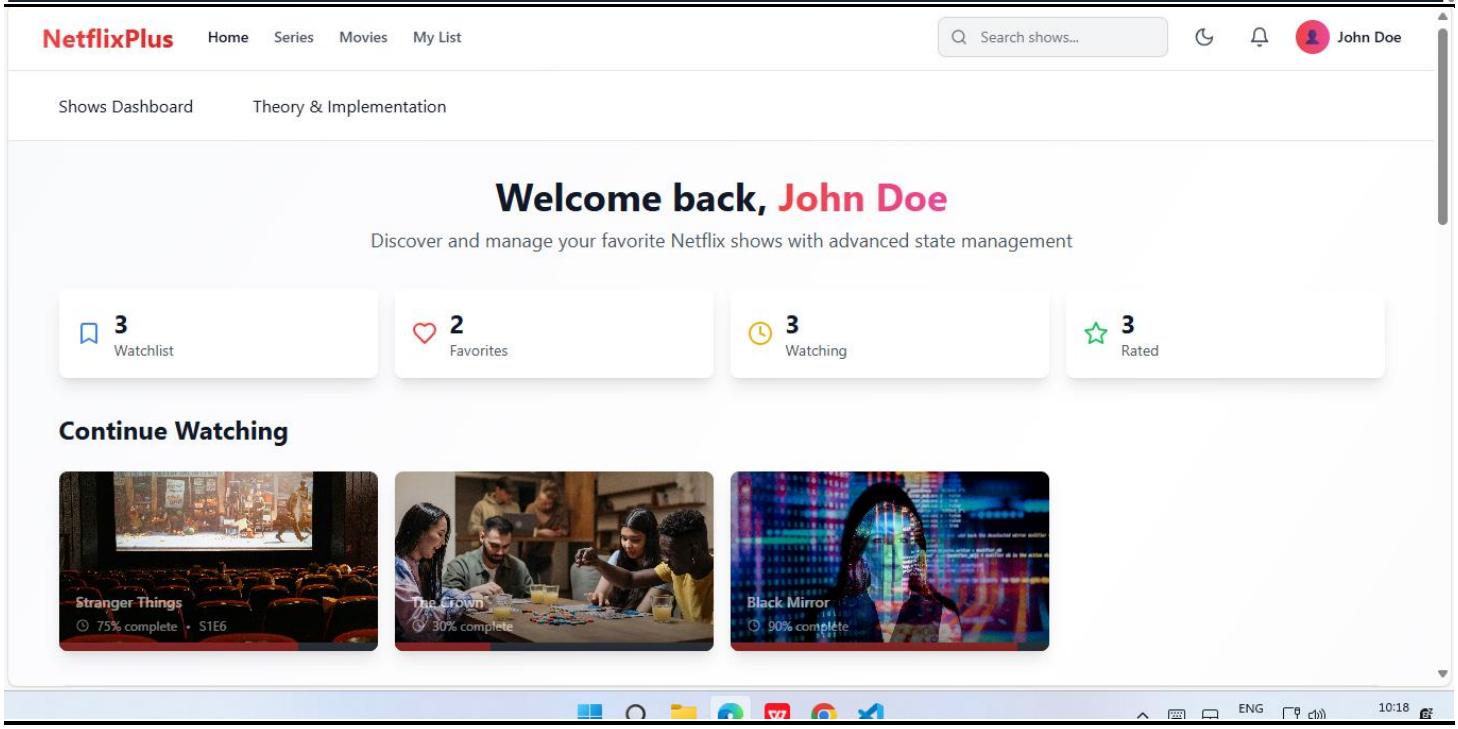
```
    year: 2020,

    rating: 8.5,

    duration: '56min',

    description: 'Orphaned chess prodigy Beth Harmon struggles with addiction while pursuing chess mastery.',

    imageUrl:    'https://images.pexels.com/photos/8111468/pexels-photo-8111468.jpeg',

    seasons: 1,

    episodes: 7,

    type: 'series',

    maturityRating: 'TV-MA',

    tags: ['chess', 'coming-of-age', 'addiction', 'female-protagonist']

  }

];



export const genres = ['All', 'Sci-Fi', 'Drama', 'Fantasy', 'Thriller', 'Crime', 'Romance', 'Comedy'];

export const maturityRatings = ['G', 'PG', 'PG-13', 'R', 'TV-MA'];
```

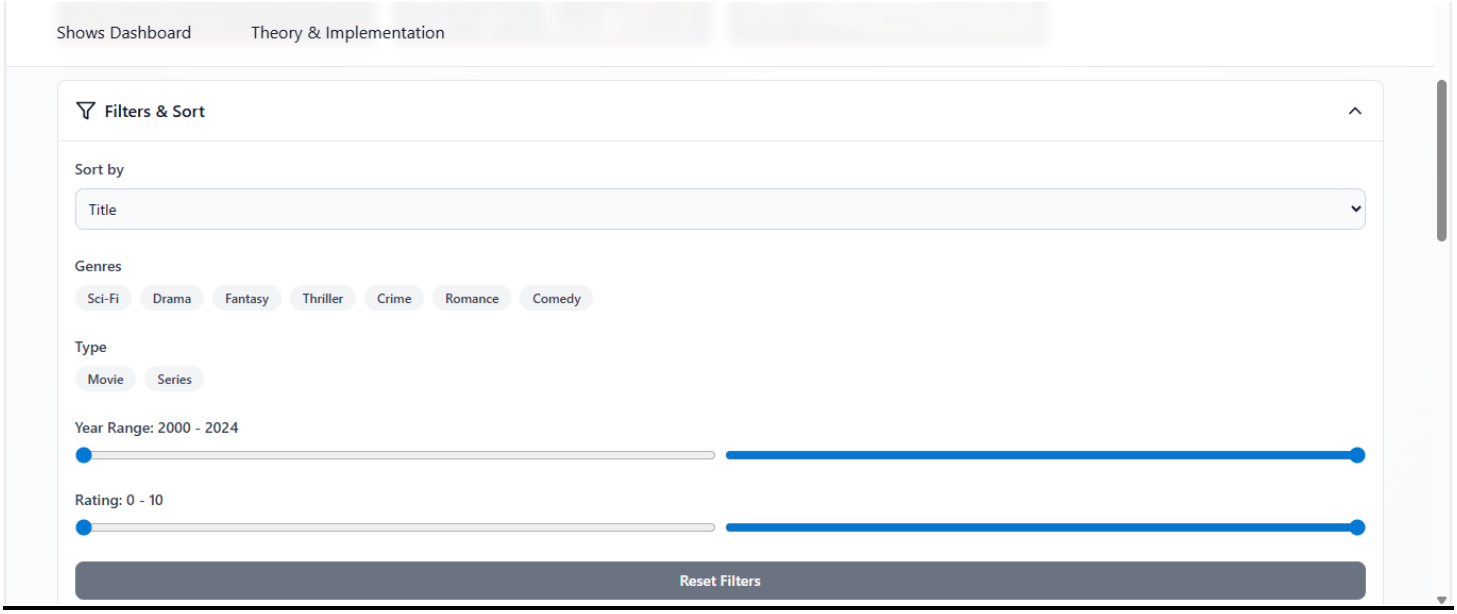
- **OUTPUT:**


localhost:5173
NetflixPlus
Home
Series
Movies
My List
Search shows...
John Doe
Shows Dashboard
Theory & Implementation
Welcome back, John Doe
Discover and manage your favorite Netflix shows with advanced state management
3
Watchlist
2
Favorites
3
Watching
3
Rated
Continue Watching
Stranger Things
75% complete • S1E6
The Crown
30% complete
Black Mirror
90% complete


NetflixPlus
Home
Series
Movies
My List
Search shows...
John Doe
Shows Dashboard
Theory & Implementation
Filters & Sort
Black Mirror
2011 • TV-MA • 6 seasons
8.8 • Your rating: 10
Rate
Sci-Fi
An anthology series exploring the dark aspects of modern technology.
dystopian
technology
psychological
Continue watching (90%)
Bridgerton
2020 • TV-MA • 3 seasons
7.3
Rate
Romance
The romantic and scandalous world of the Bridgerton family in Regency-era England.
period-drama
romance
regency
Money Heist
2017 • TV-MA • 5 seasons
8.3
Rate
Crime
An unusual group of robbers attempt to carry out the most perfect robbery in...
heist
spanish
strategy
Ozark
2017 • TV-MA • 4 seasons
8.4
Rate
Crime
A financial advisor drags his family from Chicago to Missouri Ozarks for money...
money-laundering
family
crime-drama


NetflixPlus
Home
Series
Movies
My List
Search shows...
John Doe
Shows Dashboard
Theory & Implementation
State Management Theory
Deep dive into Context API implementation and advanced patterns
Architecture
Patterns
Creativity
Performance
Multi-Context Architecture
ShowsContext
• 10 shows managed
• 10 currently filtered
• Search: "None"
• Sort by: title
• Active filters: 4
UserContext
• 2 user profiles
• 3 in watchlist
• 2 favorites
• 3 watch history items
• Theme: dark


NetflixPlus
Home
Series
Movies
My List
Search shows...
John Doe
Shows Dashboard
Theory & Implementation
Welcome back, John Doe
Discover and manage your favorite Netflix shows with advanced state management
3
Watchlist
2
Favorites
3
Watching
3
Rated
Continue Watching
Stranger Things
75% complete • S1E6
The Crown
30% complete
Black Mirror
90% complete
ENG
10:18



- **CONCLUSION:**

This experiment showcased how the Context API can effectively manage complex state in a medium-scale React application without the overhead of Redux. By implementing a multi-context architecture, reducer pattern, and custom hooks, the solution achieved clean code, modularity, and seamless cross-context communication. Advanced features like optimistic updates, user statistics, and theme management enhanced user experience. Performance was optimized through context separation, batched updates, and encapsulated logic. Overall, the approach proved scalable, performant, and well-suited for real-world, feature-rich applications.