**Documentation for React-Redux Article Viewer Application**

**Overview**

This project is a React application built with TypeScript that fetches and displays articles from a REST API. It includes state management with Redux and handles routing with react-router-dom v6. The application is designed to be visually appealing and user-friendly.

**Technologies Used**

* **React**: A JavaScript library for building user interfaces.
* **TypeScript**: A typed superset of JavaScript that compiles to plain JavaScript.
* **Redux**: A predictable state container for JavaScript apps.
* **React-Redux**: Official React bindings for Redux.
* **React Router**: A collection of navigational components that compose declaratively with your application.
* **Axios**: A promise-based HTTP client for the browser and Node.js.

**Approach**

**1. Setting Up Redux Store**

* **Store Configuration**: Configured Redux store using @reduxjs/toolkit.
* **State Management**: Created a slice for articles using createSlice from Redux Toolkit.

**2. Fetching Articles**

* **API Requests**: Used Axios to fetch articles from the provided endpoints.
* **State Updates**: Managed state updates in articlesSlice.ts.

**3. Routing**

* **React Router v6**: Implemented routing with Routes and Route components.
* **Dynamic Routing**: Utilized useParams hook to fetch specific article details.

**4. Error Handling**

* **Error Boundaries**: Created an ErrorBoundary component to catch and display errors during rendering.

**5. Components**

* **ArticleList**: Displays a list of articles fetched from the API.
* **ArticleDetail**: Displays the full text of an article based on the article ID.
* **ArticleItem**: A reusable component to display individual articles in the list.

**Challenges Faced**

**1. React Router v6 Changes**

**Issue**: The Switch component and exact prop were deprecated in v6.

**Solution**: Replaced Switch with Routes and removed exact since routes are exact by default in v6.

**2. State Management**

**Issue**: Ensuring proper state updates and avoiding unnecessary re-renders.

**Solution**: Used Redux Toolkit to simplify state management and ensure predictable state updates.

**3. Error Handling**

**Issue**: Handling errors gracefully during API calls and rendering.

**Solution**: Implemented ErrorBoundary and added error handling logic in API calls.

**Steps to Run the Application**

1. **Clone the Repository**:

git clone https://github.com/your-repo/react-redux-article-viewer.git

cd react-redux-article-viewer

1. **Install Dependencies**:

npm install

1. **Start the Application**:

npm start

1. **Open in Browser**: Open your browser and navigate to http://localhost:3000.

**Conclusion**

This project demonstrates how to build a modern React application with TypeScript, leveraging Redux for state management and React Router for navigation. It also highlights best practices for handling API calls and managing application state in a scalable manner.