Comprehensive Revision Notes: Building a Protected Route with Bearer Token Authentication and Redux Toolkit

---

Introduction

This set of notes covers the creation of a protected route within a MERN stack application. The class specifically focused on authentication using Bearer Tokens, state management using Redux Toolkit, and constructing protected components for secure route access.

Agenda

1. **Bearer Token Validation with AuthMiddleware**
2. **Axios Instance Configuration for Authenticated Requests**
3. **Building Protected Components**
4. **Implementing State Management with Redux Toolkit**

Bearer Token Validation with AuthMiddleware

In this module, we set up JWT-based authentication. The JSON Web Token (JWT) is stored in local storage and sent with requests using the `Authorization` header.

Key Concepts

- **JWT Structure**: The `Bearer` string is prefixed to the token, signaling the usage of token-based authentication.
- **Example**:

- **Middleware Creation**: Utilizes `req`, `res`, and `next` in the authentication middleware to verify the token.
- **Middleware Benefits**: Ensures token validation across different server routes like profile, bookings, and payments【4:4†source】.

Axios Instance Configuration

The Axios instance is crucial for maintaining authenticated requests across the application.

Configuration Steps

1. **Setting Headers**:

```
export const axiosInstance = axios.create({
  headers: {
    "Content-Type": "application/json",
    'authorization' : `Bearer ${localStorage.getItem('toke
  },
});
```

2. **Automatic Token Inclusion**: The Axios interceptor fetches the latest token from local storage to include in outgoing requests .
3. **Request/Response Handling**:
   - **Interceptors**: Modify request configurations before sending and handle responses before processing, giving room to include header changes like authorization .

Building Protected Components

Creating `ProtectedRoute.js`

- **Component Setup**:

```
const ProtectedRoute = ({children}) => {
  return <div>{children}</div>;
};
export default ProtectedRoute;
```

- **Integration**: Used in routing to wrap protected paths, ensuring that only authenticated users can access them .

Handling Protected Navigation

- **Role-based Navigation**:
  - If a user is an admin, navigate to the admin page; if a partner, to partner page; otherwise, to user profile page .

State Management with Redux Toolkit

The Redux Toolkit simplifies state management, particularly useful in complex applications where multiple components need to share application state.

Steps to Setup Redux

1. **Install Dependencies**:

```
npm install @reduxjs/toolkit
npm install react-redux
```

2. **Creating Slices**:
   - **LoaderSlice**: Manages loading state with actions `ShowLoading` and `HideLoading` .
   - **UserSlice**: Handles user state with an action `SetUser` .

Benefits of Redux Toolkit

- **Reduced Boilerplate**: Automatically generates actions and reducers.
- **Structured State Management**: Slices organize code by feature, simplifying maintenance .

Conclusion

This class emphasized creating a secure application utilizing middleware for authentication, Axios for handling authenticated requests, and Redux Toolkit for managing application state efficiently. Proper implementation of these concepts is vital for any robust application that demands user authentication and secure data handling.