User Authentication and Authorization Guide

Introduction

This document provides detailed instructions on how to securely integrate and manage user authentication within your applications using JWT (JSON Web Tokens). By following the steps outlined in this guide, developers can ensure that their applications maintain robust security standards, facilitate smooth user interactions, and handle user credentials and permissions effectively.

Steps

Authorization Header Requirement

To ensure secure communication, you must include the following authorization header when sending requests:

```
Authorization: Bearer YOUR_ACCESS_TOKEN_HERE
```

`Bearer` tokens are mandatory for authentication.

Decode JWT token

To extract user information from a JWT token, you can use the following code snippet:

```
import jwt from 'jsonwebtoken';
// decode your jwt
const token = 'your.jwt.token.here';
const decoded = jwt.decode(token);
console.log(decoded);
// get username
const username = decoded.username;
// get role
const userRole = decoded.role;
console.log(userRole);
```

Add Token to Every Request Globally

Ensure that the token is included in every request globally using Axios:

```
const token = 'your_access_token_here';
axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;
```

Store and Retrieve User Information

To manage user information in the browser, use the following approach:

```
localStorage.setItem('jwt', 'Bearer '+'your.jwt.token.here');
userInfo = {username:"xxx",role:"xxx"};
//localStorage only stores String，we can convert userInfo to string
localStorage.setItem('userInfo', JSON.stringify(userInfo));
const token = localStorage.getItem("token")
//Convert string to user-info
const userInfo = JSON.parse(localStorage.getItem("user-info"))
```

Logout

To log out, ensure you remove the stored token and user information from `localStorage`:

```
localStorage.removeItem('token');
localStorage.removeItem('user-info'');
```