Step 1: Setup Nuxt.js Project

First, ensure you have a Nuxt.js project set up. If not, you can create one by running:

Bash

npx create-nuxt-app my-nuxt-app

Step 2: Create Pages

Create the necessary pages:

1. **pages/login.vue**: The login page.
2. **pages/home.vue**: The home page accessible by both admin and employee.
3. **pages/admin.vue**: The admin page accessible only by admin.

Step 3: Implement Login Logic

In **pages/login.vue**, implement the login form and logic:

Vue

<template>

<div>

<form @submit.prevent="login">

<input v-model="username" type="text" placeholder="Username" />

<input v-model="password" type="password" placeholder="Password" />

<button type="submit">Login</button>

</form>

</div>

</template>

<script>

export default {

data() {

return {

username: '',

password: '',

}

},

methods: {

async login() {

try {

const response = await this.$axios.$post('/api/login', {

username: this.username,

password: this.password,

})

// Assuming response contains token and user role

this.$store.commit('setUser', response)

if (response.role === 'admin' || response.role === 'employee') {

this.$router.push('/home')

}

} catch (error) {

console.error('Login failed:', error)

}

}

}

}

</script>

Step 4: Implement Middleware for Route Protection

Create a middleware to protect routes:

**middleware/auth.js**

JavaScript

export default async function ({ store, redirect }) {

if (!store.state.user) {

return redirect('/login')

}

}

**middleware/admin.js**

JavaScript

export default async function ({ store, redirect }) {

if (store.state.user.role !== 'admin') {

return redirect('/home')

}

}

Apply these middlewares to the respective pages:

* In **pages/home.vue**, add middleware: 'auth', in the script section.
* In **pages/admin.vue**, add middleware: ['auth', 'admin'], in the script section.

Step 5: Server-Side API for Login and Role Validation

You need a server-side API to validate the user credentials and return the user details including role. Here’s a simple example using Node.js and Express:

JavaScript

const express = require('express')

const app = express()

const port = 3001

app.use(express.json())

let users = [

{ username: 'admin', password: 'admin', role: 'admin' },

{ username: 'employee', password: 'employee', role: 'employee' },

]

app.post('/api/login', (req, res) => {

const { username, password } = req.body

const user = users.find(u => u.username === username && u.password === password)

if (user) {

res.json({ token: 'some-token', role: user.role })

} else {

res.status(401).json({ message: 'Invalid credentials' })

}

})

app.listen(port, () => {

console.log(`Server running on port ${port}`)

})

This example stores passwords in plain text for simplicity, but you should hash and compare passwords securely in a real application.

Step 6: Vuex Store Setup

Ensure you have Vuex set up to store user information:

**store/index.js**

JavaScript

import { createStore } from 'vuex'

const store = createStore({

state() {

return {

user: null,

}

},

mutations: {

setUser(state, user) {

state.user = user

},

},

})

export default store