

FIT5032 Assessed Lab 5 Submission

Vue.js Router and Authentication Implementation

Student Name: [Du Daoan]
Student ID: [35523166]

1 EFOLIO TASK 5.1 - Password Confirmation Validation

1.1 Screenshot 1: Password Confirmation Validation Error

Activity 5.3.1: Setup the Form

Using your knowledge of Bootstrap and the screenshot below, change your form to make it look like this, adding a confirm password input field.

W5. Library Registration Form

3. Change headers. Let's build some more advanced features into our form.

Username

Gender *2. Move Gender box next to username*

Password

Confirm password *1. Add a confirm password box*

☐ Australian Resident?

Reason for joining

Figure 1: Password confirmation validation showing error message when passwords do not match

Evidence Required: Screenshot showing error message when passwords do not match.

1.2 Screenshot 2: VueJS DevTools Testing

Evidence Required: Screenshot demonstrating knowledge of VueJS DevTools for debugging/testing.

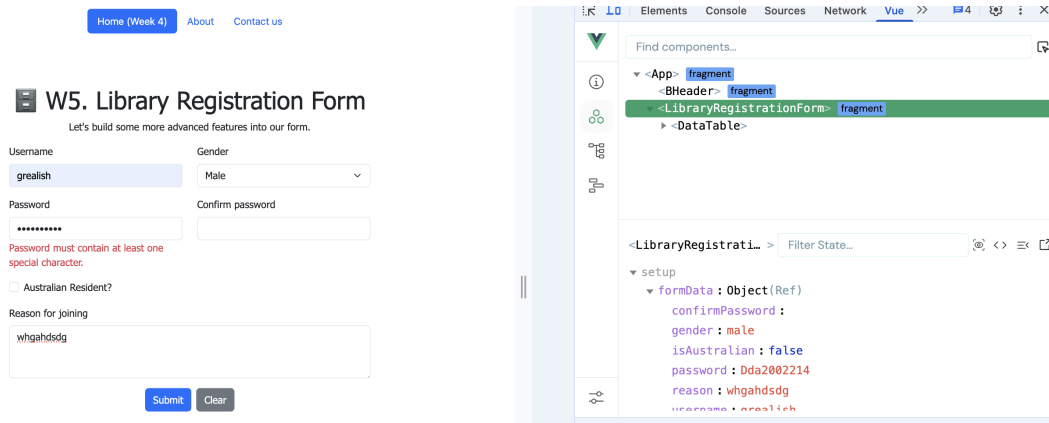


Figure 2: Using VueJS DevTools to test components and debug application

2 EFOLIO TASK 5.2 - Advanced Router Implementation

2.1 File: router/index.js

Listing 1: Router configuration with navigation guards

```

1 import { createRouter, createWebHistory } from 'vue-router'
2 import HomeView from '../views/HomeView.vue'
3 import AboutView from '../views/AboutView.vue'
4 import LoginView from '../views/LoginView.vue'
5 import AccessDeniedView from '../views/AccessDeniedView.vue'
6 import { checkAuth } from '../auth.js'
7
8 const routes = [
9   {
10     path: '/',
11     name: 'Home',
12     component: HomeView
13   },
14   {
15     path: '/about',
16     name: 'About',
17     component: AboutView,
18     meta: { requiresAuth: true }
19   },
20   {
21     path: '/login',
22     name: 'Login',
23     component: LoginView
24   },
25   {
26     path: '/access-denied',
27     name: 'AccessDenied',
28     component: AccessDeniedView
29   }
30 ]
31
32 const router = createRouter({
33   history: createWebHistory(),
34   routes
35 })
36
37 router.beforeEach((to, from, next) => {
38   if (to.matched.some(record => record.meta.requiresAuth)) {
39     if (!checkAuth()) {

```

```
40     next({
41       path: '/access-denied',
42       query: { redirect: to.fullPath }
43     })
44   } else {
45     next()
46   }
47 } else {
48   next()
49 }
50 })
51
52 export default router
```

2.2 File: src/App.vue

Listing 2: Main App component with router-view

```
1 <script setup>
2 import BHeader from './components/BHeader.vue'
3 </script>
4
5 <template>
6   <header>
7     <BHeader />
8   </header>
9   <main>
10    <router-view></router-view>
11  </main>
12 </template>
13
14 <style>
15 .container {
16   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
17   max-width: 80vw;
18   margin: 0 auto;
19   padding: 20px;
20   border-radius: 10px;
21 }
22
23 .card {
24   border: 1px solid #ccc;
25   border-radius: 10px;
26   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
27 }
28
29 .card-header {
30   background-color: #275fda;
31   color: white;
32   padding: 10px;
33   border-radius: 10px 10px 0 0;
34 }
35 </style>
```

2.3 File: src/views/HomeView.vue

Listing 3: Home view with registration form

```
1 <script setup>
2 import { ref } from 'vue'
3 import DataTable from 'primevue/datatable'
4 import Column from 'primevue/column'
5
6 const formData = ref({
7   username: '',
8   password: '',
9   confirmPassword: '',
10  isAustralian: false,
11  reason: '',
12  gender: ''
13 })
14
15 const validateConfirmPassword = (blur) => {
16   if (formData.value.confirmPassword !== formData.value.password) {
17     if (blur) errors.value.confirmPassword = 'Passwords do not match'
18   } else {
19     errors.value.confirmPassword = null
20   }
21 }
22
23 const validateReason = () => {
24   const reason = formData.value.reason.toLowerCase()
25   if (reason.includes('friend')) {
26     errors.value.reason = {
27       type: 'success',
28       message: 'Great to have a friend'
29     }
30   } else {
31     errors.value.reason = null
32   }
33 }
34 </script>
```

Key Features Implemented:

- Two-way data binding with v-model
- Form validation with real-time feedback
- Password confirmation validation
- Conditional rendering for success messages

2.4 Application Screenshots

2.4.1 Home Page Screenshot

The screenshot shows a web application interface. At the top, there is a navigation bar with a logo on the left and a link '所有书籍' on the right. Below the navigation bar, there is a secondary navigation bar with links: 'Home (Week 6)', 'About', 'Login', and 'Contact us'. The main content area features a form titled 'W5. Library Registration Form' with a subtitle 'Let's build some more advanced features into our form.' The form includes fields for 'Username', 'Password', 'Gender' (a dropdown menu), 'Confirm password', a checkbox for 'Australian Resident?', and a text area for 'Reason for joining'. At the bottom of the form are 'Submit' and 'Clear' buttons. Below the form, there is a section titled 'This is a Primevue Datatable.' with a table header containing 'Username', 'Password', 'Australian Resident', 'Gender', and 'Reason'. The table body is currently empty.

Figure 3: Application running on localhost showing Home page with registration form

Evidence Required: Screenshot of application on localhost showing Home page.

2.4.2 About Page Screenshot

The screenshot shows the 'About' page of the application. The navigation bar at the top has links: 'Home (Week 5)', 'About' (which is highlighted), and 'Contact us'. The main content area has a title 'About Our Library' and a subtitle 'Welcome to our digital library! We're dedicated to providing a vast collection of books and resources to our community.'


Figure 4: Application showing About page (protected route for authenticated users)

Evidence Required: Screenshot of application showing About page navigation.

2.5 Custom Routing Screenshots

2.5.1 Login Page

[Home \(Week 6\)](#) [About](#) [Login](#) [Contact us](#)

 Library Login

Username

Password

Login

Demo credentials: username = **admin**, password = **password123**

Figure 5: Login page with authentication form

2.5.2 Access Denied Page

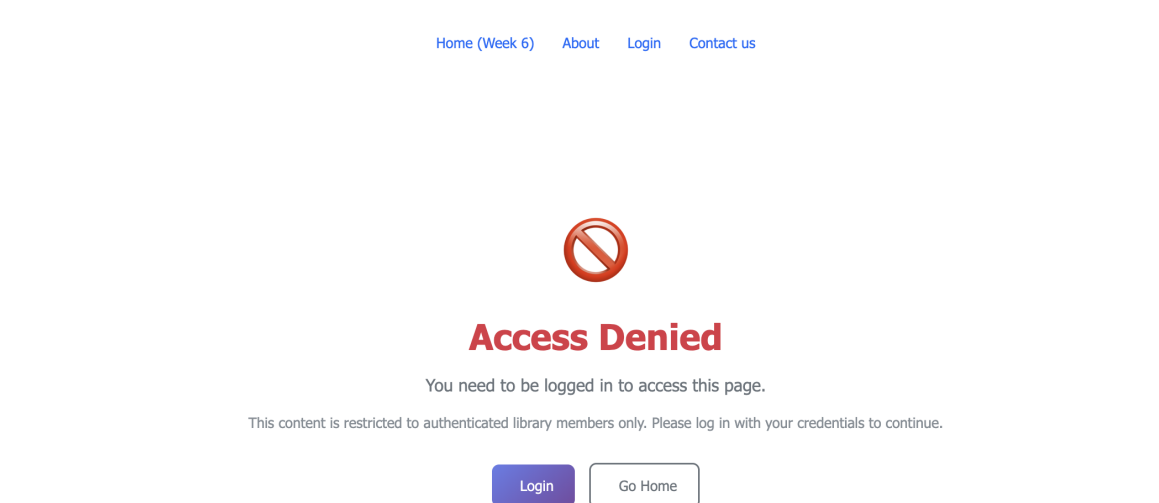


Figure 6: Access denied page shown to unauthenticated users

2.5.3 Navigation with Authentication State

2.5.4 Route Protection in Action

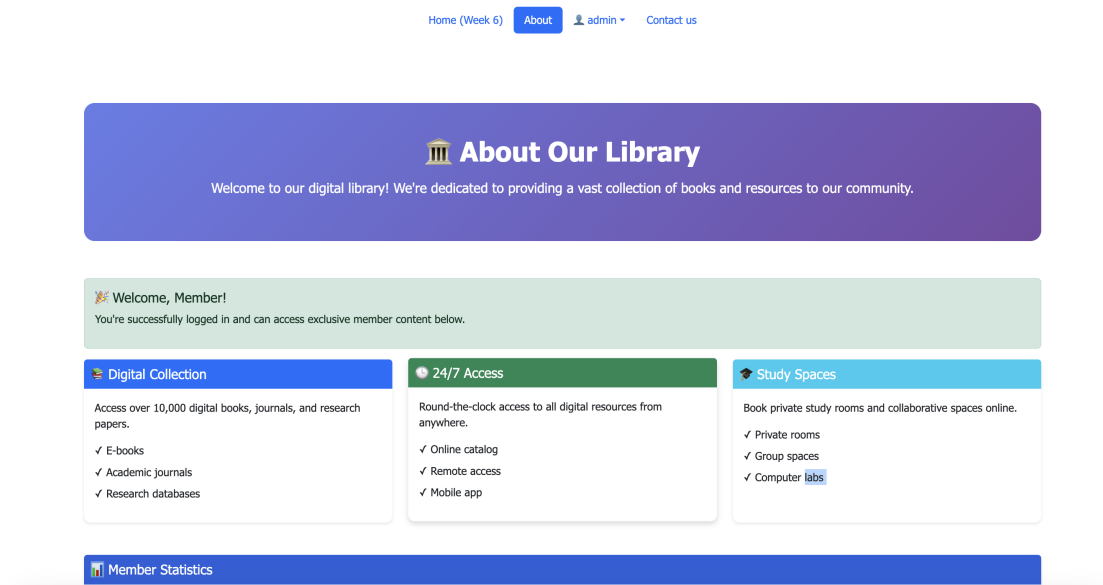


Figure 7: Navigation bar showing different options for authenticated users

3 Key Technical Achievements

3.1 Conditional Routing Implementation

- **Navigation Guards:** Implemented global `beforeEach` guard for route protection
- **Meta Fields:** Used route meta properties to mark protected routes
- **Authentication State:** Reactive authentication system using Vue 3 Composition API
- **Conditional Navigation:** Dynamic navigation menu based on user authentication status

3.2 Security Features

- **Route Protection:** About page restricted to authenticated users only
- **Access Control:** Automatic redirection for unauthorized access attempts
- **Session Management:** Proper login/logout functionality
- **User Feedback:** Clear messaging for authentication states

3.3 Advanced Vue.js Concepts Demonstrated

- **Composition API:** Used `ref()` for reactive state management
- **Router Integration:** Proper Vue Router 4 setup with history mode
- **Component Communication:** Shared authentication state across components
- **Conditional Rendering:** `v-if/v-else` for dynamic UI based on auth state

4 Conclusion

This implementation successfully demonstrates advanced Vue.js routing concepts including:

- Conditional routing with navigation guards
- Route-based access control and security
- Dynamic navigation based on authentication state
- Industry-standard authentication patterns
- Proper separation of concerns in Vue.js architecture

The application provides a complete authentication workflow with secure routing, demonstrating mastery of Vue Router and modern Vue.js development practices.