6 Months Training TR-104 Full Stack Development

January, 2026

Day 8 Report

The eighth day of training focused on completing a full frontend authentication and authorization flow in an Angular application using the NgModule-based architecture. The session emphasized enforcing login rules, implementing role-based access control, enhancing UI behavior based on authentication state, and strengthening form validation using custom validators. The day involved hands-on implementation, architectural reasoning, and deep understanding of Angular routing, guards, services, and Reactive Forms.

Authentication Enforcement Using Route Guards

A major focus of the day was enforcing authentication and authorization at the routing level using Angular route guards. Two types of guards were implemented:

- **Authentication Guard (AuthGuard)**

  Ensured that only logged-in users could access protected routes such as '/books' and '/admin'. Unauthorized users attempting to access these routes were automatically redirected to the login page.
- **Authorization Guard (AdminGuard)**

  Enforced role-based access control by allowing only users with the 'admin' role to access the '/admin' route. Non-admin users were redirected to the books page.

These guards were applied at the routing level using the "canActivate" property, ensuring that route access decisions were made before component initialization.

Role-Based Access Control

User roles were managed using a centralized authentication service. Based on mock login logic:

- Users logging in with the username ‘admin’ were assigned the ‘admin’ role
- All other users were assigned the ‘user’ role

This role information was used consistently across guards and UI logic to control access and visibility without duplicating authentication logic in components.

UI Enforcement Based on Authentication State

The application’s navigation UI was enhanced to dynamically respond to authentication and authorization state:

- The ‘Login’ link was hidden once the user logged in
- A ‘Logout’ button was displayed for authenticated users
- The ‘Admin’ navigation link was displayed only for users with the admin role
- Logout functionality cleared authentication state and redirected the user to the login page

These changes improved usability and ensured that users were presented only with relevant navigation options while still relying on guards for actual security enforcement.

Password Strength Validation Using Custom Validators

The login form was further enhanced by implementing strong password validation using a custom Angular validator. In addition to basic validation rules, the password field was validated to ensure it contained:

- At least one uppercase letter

- At least one lowercase letter
- At least one numeric character
- At least one special character
- A minimum length requirement

A custom validator function was created and attached to the password form control. Specific error messages were displayed dynamically in the UI based on which password rules were violated. This demonstrated real-world form validation practices and advanced usage of Reactive Forms.

Reactive Forms and Validation Handling

The login form continued to use Angular Reactive Forms, with validation logic fully defined in TypeScript. Key aspects included:

- Use of built-in validators such as ‘required’, ‘minLength’, and ‘email’
- Integration of a custom password strength validator
- Display of validation messages based on form control state
- Automatic disabling of the submit button when the form was invalid

This approach ensured predictable form behavior, strong validation, and clean separation between form logic and presentation.

Key Learnings

- Learned how to enforce authentication using Angular route guards
- Understood the difference between authentication and authorization
- Implemented role-based access control using guards and services
- Learned how to control UI visibility based on login and role state

- Implemented logout functionality with proper state cleanup
- Gained hands-on experience creating and using custom form validators
- Learned how to enforce real-world password strength rules
- Strengthened understanding of Angular routing, services, and Reactive Forms
- Improved confidence in designing secure frontend application architecture

Conclusion

The eighth day of training focused on completing a robust and realistic authentication system in an Angular application. By enforcing login rules with route guards, implementing role-based access control, enhancing UI behavior, and adding strong password validation, participants gained practical experience with professional Angular development practices. This session solidified core concepts related to security, routing, validation, and application architecture, providing a strong foundation for upcoming features such as state persistence, admin panel development, and core application functionality.