6 Months Training TR-104 Full Stack Development

January, 2026

Day 7 Report

The seventh day of training focused on implementing a proper authentication foundation in an Angular application using the NgModule-based architecture. The session emphasized completing a realistic login flow using Reactive Forms, understanding validation mechanisms, dependency injection, and TypeScript strict type checking. The day involved extensive hands-on practice, debugging, and deep conceptual clarification to strengthen both Angular and TypeScript fundamentals.

Authentication Setup Using Reactive Forms

The primary task of the day involved designing and implementing a structured login system. The login feature was upgraded from a basic form to a professional Reactive Forms-based implementation. The login form was designed to include the following fields:

- Username
- Email
- Password

Reactive Forms were used to ensure that form structure, validation rules, and behavior were defined in TypeScript rather than in the template. This approach improved predictability, maintainability, and scalability of the authentication logic.

Form Validation Implementation

Proper validation rules were implemented for all login fields to ensure data correctness and user feedback:

- Username validation:
    - Mandatory field
    - Minimum length requirement
- Email validation:
    - Mandatory field
    - Valid email format check
- Password validation:
    - Mandatory field
    - Minimum length requirement

Angular's built-in validators were used to enforce these rules, and conditional error messages were displayed based on form control state (touched and invalid). The submit button was disabled automatically when the form was invalid, ensuring that incomplete or incorrect data could not be submitted.

Understanding Dependency Injection

A major conceptual focus of the day was understanding Dependency Injection in Angular. The following aspects were covered in detail:

- Meaning of dependencies and injection
- Role of constructors in dependency injection
- How Angular creates and provides shared services
- Why services should not be instantiated manually using the new keyword

- Use of the @Injectable decorator to make services injectable

This understanding clarified how the authentication service and router were injected into the login component and shared across the application.

Authentication Service Design

An authentication service was created to act as a central store for login state and user information. The service:

- Stored login status
- Stored user details (username, email, role)
- Assigned roles (admin or user) using mock logic
- Exposed data safely using getter functions

This approach separated authentication logic from UI components and established a single source of truth for user state within the application.

TypeScript Strict Mode and Template Errors

An important part of the day involved debugging and understanding a TypeScript template error related to Reactive Forms. The error occurred due to strict type checking when accessing form controls using dot notation. This led to a detailed explanation of:

- Index signatures in TypeScript
- Why “FormGroup.controls” is treated as a dynamic object
- The difference between dot notation and bracket notation
- Why bracket notation is required in Angular templates under strict typing

This clarified that the issue was related to TypeScript’s type system rather than Angular itself.

Key Learnings

- Learned how to build authentication forms using Reactive Forms
- Understood how Angular validation works at both form and control levels
- Gained a strong understanding of Dependency Injection in Angular
- Learned how to design and use a centralized authentication service
- Understood the purpose and use of getter functions
- Gained exposure to TypeScript strict type checking in Angular templates
- Learned how index signatures affect property access in TypeScript
- Improved confidence in debugging Angular and TypeScript errors
- Strengthened overall understanding of Angular application flow

Conclusion

The seventh day of training focused on building a solid authentication foundation while strengthening core Angular and TypeScript concepts. By implementing Reactive Forms with proper validation, understanding dependency injection, and resolving strict template typing issues, participants gained practical experience with professional Angular development practices. This session established a strong base for implementing route guards, role-based access control, and secure navigation in upcoming sessions.