6 Months Training TR-104 Full Stack Development

January, 2026

Day 9 Report

The ninth day of training focused on configuring Firebase integration and strengthening Angular application architecture using the NgModule-based approach. The session emphasized modular routing, lazy loading, asset management in Angular 18, and feature-level routing setup. Extensive hands-on work was carried out to stabilize configuration changes, resolve routing issues, and ensure proper UI rendering within a structured Angular project.

Firebase Project and Angular Integration Setup

The primary activity of the day involved integrating Firebase with the Angular application. Tasks included:

- Creating a Firebase project
- Enabling Email/Password authentication
- Integrating Firebase into the Angular application using AngularFire
- Handling compatibility considerations between Angular 18 and Firebase libraries
- Successfully configuring Firebase services without runtime errors

This established a working Firebase-enabled Angular application environment.

NgModule-Based Lazy Loading Implementation

Lazy loading was implemented to improve application structure and performance. The following steps were completed:

- Understanding and applying lazy loading concepts
- Configuring ‘loadChildren’ with dynamic ES module imports
- Lazy loading feature modules such as:
    - Authentication module
    - Dashboard module
- Verifying that feature modules load only when their respective routes are accessed

This reinforced modular application design using NgModules.

Feature Routing Configuration

Feature-level routing was introduced and configured to support lazy-loaded modules. Activities included:

- Creating a dedicated routing module for the authentication feature
- Applying ‘RouterModule.forRoot()’ and ‘RouterModule.forChild()’ correctly
- Understanding the routing hierarchy between root and feature modules
- Resolving a blank-page issue caused by a missing routing module import
- Verifying authentication routes such as ‘/auth’, ‘/auth/sign-in’, and ‘/auth/sign-up’

These steps ensured proper navigation and route resolution across the application.

Angular 18 Asset Management and UI Integration

The session also addressed asset handling changes introduced in Angular 18. Completed tasks included:

- Understanding Angular 18’s use of the public/ directory for static assets
- Updating angular.json to correctly map assets
- Debugging and resolving image loading issues

- Successfully rendering images on the Welcome page
- Enhancing UI using Bootstrap and Angular Material components

This ensured correct static resource loading and improved visual presentation.

Key Learnings

- Learned how to integrate Firebase with an Angular 18 application
- Understood AngularFire configuration and usage
- Gained clarity on NgModule-based lazy loading using dynamic ES module imports
- Learned the difference between ‘RouterModule.forRoot()’ and ‘RouterModule.forChild()’
- Understood the necessity of feature routing for lazy-loaded modules
- Improved ability to debug routing and module configuration issues
- Learned Angular 18 asset handling and resolved static resource problems
- Strengthened understanding of scalable Angular architecture

Conclusion

The ninth day of training focused on stabilizing Firebase integration and improving Angular application architecture through lazy loading, feature routing, and proper asset management. By resolving routing and configuration issues and ensuring correct UI rendering, the application was successfully structured using NgModule-based best practices.