6 Months Training TR-104 Full Stack Development

January, 2026

Day 6 Report

The sixth day of training focused on understanding the foundational architecture of Angular using the NgModule-based approach. The session emphasized building strong conceptual clarity rather than rushing into feature development. Key topics included Angular application structure, entry points, modules, components, decorators, module visibility rules, and component interaction. The day involved hands-on debugging, architectural correction, and gradual learning through practical tasks to ensure a solid understanding of Angular fundamentals.

Angular Project Architecture Setup

The primary activity of the day involved setting up and stabilizing an Angular project using the NgModule-based architecture, as instructed by the mentor. Since modern Angular versions default to standalone architecture, special attention was given to converting and maintaining the project strictly using NgModules.

Key architectural steps included:

- Configuring the application to bootstrap using AppModule
- Understanding the role of main.ts as the application entry point
- Creating and configuring AppModule as the root module
- Introducing a feature module (BooksModule) to represent a functional section of the application
- Ensuring correct module imports and declarations

Understanding Components and Modules

A major portion of the day was dedicated to understanding the relationship between components and modules. The following concepts were covered in detail:

- Every component belongs to exactly one module
- Components are declared inside modules using the declarations array
- Modules act as boundaries that control component visibility
- Components are private to their module by default
- Components must be explicitly exported if they are to be used by other modules

This understanding helped clarify why Angular throws errors such as “not a known element” when module rules are violated.

Public vs Private Components (NgModule Encapsulation)

The concept of module encapsulation was introduced and practiced through hands-on tasks. The distinction between private (internal) and public (exported) components was clearly established:

- Internal components (used only within the same module) do not require export
- Components used across module boundaries must be exported from their module
- The consuming module must import the module that exports the component

This was practically demonstrated by exporting BookListComponent from BooksModule to make it accessible in AppComponent.

Decorators and Angular Metadata

The session also covered Angular decorators, focusing on both conceptual and technical understanding. Topics included:

- Purpose of decorators in Angular
- Technical definition of decorators in TypeScript
- Role of @Component and @NgModule decorators
- How decorators attach metadata used by Angular during compilation
- Difference between class logic and framework configuration

This helped in understanding how Angular recognizes components and modules at compile time.

Component Composition Using Templates

Another important concept covered was component composition through HTML templates. It was clarified that:

- Components are nested and used via their selectors in HTML
- No TypeScript imports are required between components for UI composition
- Angular resolves component usage through module declarations and exports
- Templates, not ".ts" files, define component nesting and structure

This was reinforced by creating and using a secondary component (BookDetailsComponent) inside the template of BookListComponent.

Error Handling and Debugging

The day also involved understanding and handling common Angular errors, including:

- Errors related to missing exports
- Errors caused by incorrect module visibility
- Understanding startup error handling using ".catch(err => console.error(err))" during application bootstrap
- Interpreting Angular compiler and runtime error messages

This hands-on debugging improved confidence in diagnosing and fixing architectural issues.

Key Learnings

- Understood Angular NgModule-based application architecture
- Learned the role of “main.ts” and “AppModule” in bootstrapping the app
- Gained clarity on how components belong to and are scoped within modules
- Learned why components are private by default in Angular
- Understood the purpose of “exports” and “imports” in NgModules
- Learned how components interact through HTML templates
- Gained technical understanding of decorators and metadata
- Practiced debugging common Angular module and component errors
- Developed a strong conceptual foundation for further Angular development

Conclusion

The sixth day of training focused on strengthening Angular fundamentals by emphasizing architectural clarity and hands-on learning. Instead of rushing into advanced features, the session ensured a deep understanding of how Angular applications are structured using NgModules, how components are scoped and shared, and how Angular uses decorators and metadata to function. This foundational knowledge is critical for building scalable, maintainable applications and will serve as a strong base for upcoming topics such as routing, services, and reactive forms.