Lecture 12: Intro to Angular and Architecture

What is Angular?

- **Angular** is a comprehensive framework for building dynamic web applications.
- Developed by Google, it leverages TypeScript, RxJS, and powerful CLI

High-Level Angular Architecture

- **Modules –** Organize the application into cohesive blocks
- **Components –** Define views and encapsulate UI behavior
- **Templates –** HTML with Angular directives and bindings
- **Services –** Business logic, reusable functions shared across components
- **Dependency Injection –** Provides instances of services to components and modules

Building Blocks of Angular Applications

- Angular Modules (NgModules)
- Components
- Templates and Directives
- Services and Dependency Injection
- Routing Module (optional)

Angular Modules

Organizes code in cohesive blocks.

Types of Modules:

- **Root Module –** Bootstraps the application, typically **AppModule**
- **Feature Modules –** Organize related code, like **UserModule, AdminModule**
- **Shared Module –** Contains common functionality, reusable components, pipes, and directives

Advantages:

- Code organization and maintainability
- Lazy loading for performance optimization

Components

The building block for Angular's UI

Structure:

- **HTML Template –** Defines the component's view
- **CSS Styles –** Styling specific to the component
- **TypeScript Class –** Contains properties and methods to control the view
- **Metadata –** Configures the component using the **@Component** decorator

Component Communication:

- **@Input()** and **@Output()** – Share data and events between parent and child components.

Templates and Data Binding

Templates:

- Define HTML stucture and use Angular syntax for dynamic content
- Use structural and attribute directives

Data Binding:

- **Interpolation { { } } –** Embed dynamic data
- **Property Binding [property] –** Bind a property to an expression
- **Event Binding (event) -** Trigger actions from user events
- **Two-Way Binding [(ngModel)] –** Sync data between model and view

Directives:

- **Structural - *ngIf *ngFor -** Change DOM structure
- **Attribute - ngClass ngStyle -** Change DOM appearance or behavior

Services and Dependency Injection (DI)

Services:

- Provide business logic, data management, or helper functions
- Use **@Injectable( )** decorator to indicate an injectable class

Dependency Injection:

- Design pattern to inject dependencies (services) where needed

Angular Binding

Router Module:

- Navigates between views
- Defines routes in an application with route paths and associated components

Features:

- **Route Guards –** Control access to routes.
- **Lazy Loading –** Loads specific feature modules on demand
- **Parameter Passing –** URL parameters for dynamic routing

Angular Forms

Template-Driven Forms:

- Easy to create, suitable for simple forms

Reactive Forms:

- **FormControl** and **FormGroup** for complex forms
- Offers greater control, especially for validation and form state tracking