1- What is Angular ?

- Angular is a JavaScript binding framework which binds the HTML UI and JS model.
- Angular is a popular open-source web application framework maintained by Google and a community of developers. It's used for building dynamic web applications, particularly single-page applications (SPAs).
- This helps you to reduce effort on writing those lengthy lines of code for binding.
- It also has features like routing by which you can build SPA ,HTTP, DI and a nice build engine like Angular CLI.

2- What is the use of Components and Modules ?

Components:

Components are the basic building blocks of an Angular application's user interface. Each component typically represents a specific feature or section of the application. Components help in creating reusable and maintainable code by breaking the UI into smaller, manageable pieces.

Module:

Modules in Angular are used to organize the application into cohesive blocks of functionality (مجموعات متماسكه من الوظائف). Each Angular application has at least one module, known as the root module (usually named AppModule), which is responsible for bootstrapping (تمهيد)the application. Modules can also be used to group related components, directives, pipes, and services into feature modules, which can then be imported into the root module or other feature modules.

3- Explain SPA , Single Page Application ?

A Single Page Application is a web application that dynamically rewrites the current page rather than loading entire new pages from the server.Single page application are applications where the main UI gets loaded once and then the needed UI is loaded on demand.(Not Reload)
Angular is particularly well-suited for building SPAs because of its features such as:
Component-based architecture, Routing, HTTP Client, Dependency Injection.

ازاى بيتم reload وازاى بمنع reload دى يتم ؟

1- Reload: هو انك كل مرة بتطلب فيها request بيرد عليك response (HTML +CSS +JS) للصفحة المطلوب وفى كل مرة بتعمل فيها request بيعمل reload علشان يحمل response (HTML +CSS +JS) للصفحة المطلوبة .

2- ازاى همنع reload: هو انى هبعت كل (HTML +CSS +JS) بتاع كل الصفحات بتاع الموقع بس هظهر اللى انا عايز بس (SPA).

4- Explain the importance of routing and how to implement the same in Angular?

Routing in Angular is a fundamental concept (Mechanism) that allows you to create Single Page Applications (SPAs) with multiple views and navigate between them without the need to reload the entire page. It helps in organizing the application's UI into logical views.
In your app.component.html file, add the <router-outlet></router-outlet> tag. This tag acts as a placeholder where the component corresponding to (مطابق لى) the current route will be rendered (سوف يتم عرضه).
Use the routerLink directive in your templates to navigate between routes.
Redirects and Wildcard Routes: You can configure redirects and wildcard routes to handle undefined routes or redirect to a default route.

```
{ path: '404', component: PageNotFoundComponent },
{ path: '**', redirectTo: '/404' }
```

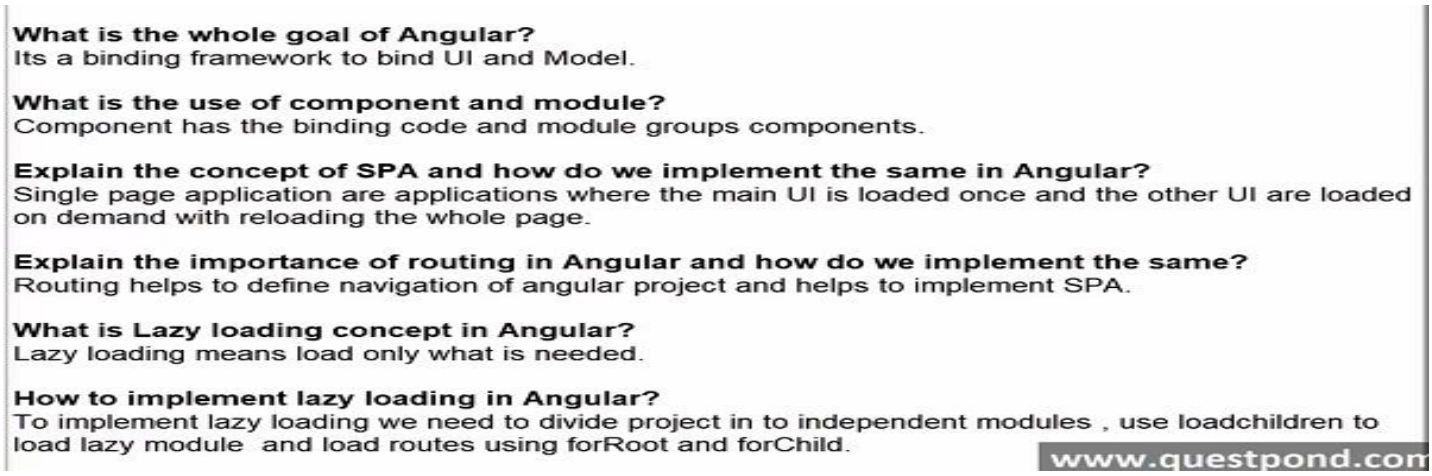
```
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
```

5- What is Lazy loading concept in Angular ?

is a technique used to load feature modules only when they are requested, rather than loading them all at once when the application starts. This helps in reducing the initial bundle size of the application, which can lead to faster load times and improved performance.
Lazy loading is achieved by using the loadChildren property in the Angular router configuration.

6- How to implement lazy loading in Angular?

1. Divide project in to seperate modules.
2. User "loadChildren" for loading the modules lazily
3. use "forRoot" to load the routes of the main module and "forChild" to load the child modules.



7- Difference between AngularJS vs Angular?

	AngularJS (1.x)	Angular (2,4,5,6,…,17)
Year	2010	2014
Language	JavaScript	typescript
Architecture	(VMC)Controller	Component-based
Mobile compliant (متوافق)	No	Yes
CLI	No	Yes
Lazy loading	No	Yes

8- What are directives in Angular?

- Directives in Angular are markers on a DOM element that tell Angular's HTML compiler ($compile) to attach a specified behavior to that DOM element or even transform the DOM element and its children.
- Directives change the behavior of HTML DOM.

9- Explain the different types of Angular directives ? SAC (Structural – attribute - component)

A. Structural Directives: Change the DOM layout by adding and removing elements.

B. **Attribute directives:-** Change the appearance and behavior of HTML elements.
C. **Component Directives:** These are the most common type of directive used in Angular applications. Components are directives with a template.

10-Explain the importance of NPM and Node Modules folder ?

NPM is a package manager which makes installation of Javascript framework easy.
"node_modules" is the folder where all the packages are installed.

11- Explain the importance of Package.json file in Angular ?

It has all the Javascript references needed for a project. So rather than installing one package at a time we can install all packages in one go (npm install).

- **Dependency Management**: It lists all the dependencies (both runtime and development) of your Angular project. This includes Angular itself, as well as any third-party libraries (like Angular Material, Toastr, etc.) and tools (like TypeScript, Angular CLI, etc.) your project relies on.
- **Version Control**: Since package.json contains a list of dependencies, it is typically version controlled using a version control system like Git. This allows you to track changes to dependencies over time, making it easier to roll back to a previous state if needed or to collaborate with others on the project.
- **Scripts**: package.json can also contain scripts that can be run using npm or yarn. These scripts can be used to automate(لأتمتة) common tasks like building the project, running tests, starting a development server, etc.
- **Metadata**: package.json can contain metadata about the project, such as its name, version, description, author, license, etc. This information is useful for developers and users of the project to understand its purpose and usage.

12- What is typescript and why do we need it?

TypeScript is a programming language developed and maintained by Microsoft. It is a superset of JavaScript, which means that any valid JavaScript code is also valid TypeScript code. TypeScript adds optional static typing to JavaScript, which can help catch errors at compile time (static type checking) and improve the overall quality of your code.

Why?

1. **Static Typing:** TypeScript allows you to specify types for variables, function parameters, and return values. This helps catch type-related errors early in the development process, before the code is even executed.
2. **Improved Code Quality:** TypeScript makes the code more self-documenting (التوثيق الذاتي) and easier to understand.
3. **Enhanced Developer Productivity:** TypeScript's static typing can help developers write code faster by providing autocompletion and type checking as they write code. This can reduce the time spent debugging and fixing errors later in the development process.

13- Explain importance of Angular CLI?

1. The Angular CLI (Command Line Interface) is a powerful tool that helps developers create, build, test, and deploy Angular applications more efficiently.
2. The Angular CLI provides a streamlined (مبسطة) way to create new Angular projects. It sets up the project structure, configuration files, and initial dependencies, saving developers time and ensuring consistency (الاتساق)across projects.
3. Angular CLI provides commands to generate components, services, modules, directives, and other Angular artifacts
4. The CLI includes a built-in development server that can be used to run and test Angular applications locally. This allows developers to quickly see their changes in action without having to set up a separate server.
5. The CLI integrates seamlessly with other Angular features and tools, such as Angular Material, Angular Universal, and Angular libraries. This makes it easier to add these features to an Angular project and leverage their capabilities.

14-What is a decorator in Angular ?

- decorator defines what kind of Angular class is it.
- In Angular, a decorator is a special kind of declaration that can be attached to a class declaration, method, accessor, property, or parameter. Decorators are used to provide metadata about the class or member they are attached to.
- Decorators can be used for a variety of purposes in Angular, such as:
    1. **Component Configuration:** The @Component decorator is used to define an Angular component. It provides metadata about the component, such as its selector, template, styles, and more.
    2. **Module Configuration**: The @NgModule decorator is used to define an Angular module. It provides metadata about the module, such as its imports, declarations, providers, and bootstrap components.
    3. **Dependency Injection:** The @Injectable decorator is used to define a service in Angular that can be injected into other components or services. It tells Angular that the class can be used as a dependency.
    4. **Input and Output Properties**: The @Input and @Output decorators are used to define input and output properties on Angular components.

15-What are Annotationa (التعليقات التوضيحية) or MetaData ?

annotations or metadata are used to decorate classes, methods, and properties to provide additional information to the Angular framework. Annotations in Angular are achieved using decorators, which are functions that add metadata to classes, methods, or properties.

`@Component ===>` selector, template, styles, and more. (component)

`@Directive ===>` selector and more.(directive)

`@Injectable ===>` selector, template, styles, and more.(service).

`@Input and @Output ===>`Used to define input and output properties of a component or directive.

16-what is a template in angular?

template is a part of a component that defines how the component's view should be rendered. It is typically written in HTML with Angular-specific syntax and can include data binding, directives, and other Angular features.

Templates in Angular are used to:

- Define the structure of the component's view.
- Bind data from the component's class to the view.
- Respond to user actions through event binding.
- Display dynamic content based on conditions or loops using structural directives.

there are two ways of defining Template one is Inline (templete)and other is a seperate(templeteURL)HTML file.

Angular templates are powerful and flexible, allowing developers to create dynamic and interactive user interfaces for their applications.

17-Explain the four types of Data bindings in Angular?

Data binding is a core concept in Angular and other modern web frameworks that establishes a connection between the application's data and the user interface (UI). It allows you to synchronize data between the model (component) and the view (template).

there are four types of data binding:

1- **Interpolation ({{ }}):** It is a one-way data binding from the component to the view. You can embed expressions in double curly braces {{ }} in the HTML template. `<p>{{ greeting }}</p>`

2- **Property binding ([ ]):** It is a one-way data binding from the component to the view, where you bind the value of a component property to an element property or directive property in the template. It is used to set properties of HTML elements or Angular directives `<img [src]="imageUrl">`.

- **ClassBinding**:add or remove class by condtion:

```
  username:boolean=true /* in ts file */

  <div [class.text-danger]="username">test</div>
  <div [class]="username?'text-danger' : 'text-sccess'">test</div>
  <div [class]="{'text-success':username ,'text-danger':!username}">test</div>
```

- **StyleBinding :**

```
mycolor:string="green"/* in ts file */
<div [style.color]="mycolor">green</div>
<div [style.color]="username? 'red':'orange' ">green</div>
<div [style]="{color:'blue' ,backgroundColor:'red' , height:'300px' ,width:'500px'}
">green</div>
```

3- **Event binding (( )):** It is a one-way data binding from the view to the component, where you bind an event (like click, change, keyup, etc.) to a method in the component. When the event is triggered, the associated method is executed.

```
<button class="btn btn-danger" (click)="onButtonClick()">click binding</button> in templete
  onButtonClick(){ in ts file
    console.log('click fired')
  }
```

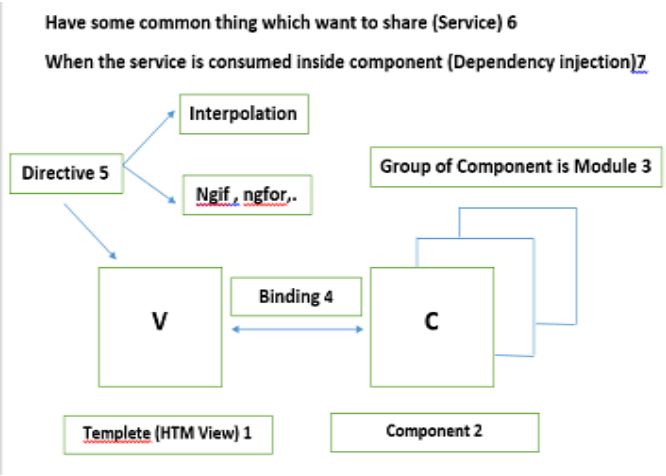
4- **Two-way binding ([( )]):** It is a combination of property binding and event binding. It allows data flow in both directions, from the component to the view and from the view to the component.

```
<input type="text" [(ngModel)]="username">     {{username}}
```

18- Explain architecture of Angular?

There are 7 important pillars in Angular Architecture.

1. Template:- The HTML view of Angular.
2. Component:- Binds the View and Model.
3. Modules:- Groups components logically.
4. Bindings :- Defines how view and component communicate.
5. Directive :- Changes the HTML DOM behaviour.
6. Services :- Helps to share common logic across the project.
7. DI :- Dependency injection helps to inject instance across constructor.



19-What are Services?

a service is a class that encapsulates reusable functionality and data that can be shared across components. Services are used to keep components lean and focused on their primary responsibilities,
Services are typically used for tasks such as fetching data from a server, logging messages, or implementing application-wide features.

20-what is dependence injection?

Dependency injection (DI) is a design pattern and a core concept in Angular (and many other frameworks) that helps to create more maintainable and modular code by promoting loose coupling (تعزيز الاقتران غير المحكم) between components. In Angular, DI is implemented by the framework itself and is used to provide dependencies (like services) to components, directives, and other services.

21-what is different between ng serve and ng build?

ng serve is used for local development and testing, while ng build is used to build the production-ready version of your application.

22- Service is stateful or stateless?

Stateless Services:

Stateless services do not maintain any internal state between calls. They typically perform operations that do not depend on previous interactions or stored data within the service itself. These services often provide utility functions, make HTTP requests, or perform calculations.

Stateful Services

Stateful services, on the other hand, maintain internal state and data between calls. They store information in properties and can provide methods to modify and access this state. These services are often used for managing user sessions, caching data, or maintaining application state.

Angular services are typically **singleton** by default when provided in the root injector (using providedIn: 'root'). This means that there is only one instance of the service throughout the application, making it easy to maintain state within the service. However, services can also be provided at different levels (e.g., component or module level), which can affect their statefulness and scope.

23-What's the difference between forChild() and forRoot ()?

In Angular, forRoot and forChild are used to configure and provide services in a way that is specific to the Angular Router and other modular features. (ModuleWithProviders<RouterModule>:

```
return{ngModule:moduleName , providers: [{provide: SomeConfig, useValue: config }]
```

forRoot():

- **Purpose**: Used to configure and initialize services and providers that are intended to be (تم تصميمهم) singletons across the entire application.
- **Usage**: Typically called in the root module (AppModule) of your application.

forChild():

Purpose: Used to configure and provide services that are scoped to (تم تحديد نطاقها) feature (تمييز) modules or sub-modules within the application.

Usage: Typically called in feature modules or lazy-loaded modules that need to define their own routes.

24- What's the difference between viewChild VS contentChild?

In Angular, @ViewChild and @ContentChild are decorators used to query(استفسار او استعلام) and interact with elements in the component's view and projected content

@ViewChild:

Purpose: @ViewChild is used to access a child component, directive, or DOM element from the view (template) of the current component.

Usage:

- Typically used to interact with elements that are part of the component's own template.
- Useful for manipulating DOM elements directly, calling methods on child components, or accessing directive properties.

Lifecycle Hook: The @ViewChild queries are resolved after the component's view has been initialized. Thus, ngAfterViewInit is the appropriate lifecycle hook to use.

@ContentChild:

Purpose: @ContentChild is used to access a child component, directive, or DOM element that has been projected (تم عرضه)into the component's view using content projection (e.g., <ng-content>).

Usage: Typically used to interact with elements that are projected into the component from the parent component's template.

Lifecycle Hook: The @ContentChild queries are resolved after the component's projected content has been initialized. Thus, ngAfterContentInit is the appropriate lifecycle hook to use.

25- Why do you use providedIn inside the services?

In Angular, the providedIn property within the @Injectable decorator is used to specify (تحديد) the provider scope for a service. This property determines where and how the service will be instantiated(إنشاء) and provided within the application. The most common value for providedIn is 'root', which configures (تكوين) the service to be a singleton and available throughout the entire(بالكامل) application.

Why?

Tree shaking(الشجرة المتزعزه): Optimizes the final bundle by excluding(استبعاد) unused services.

Singleton services: Ensures a single instance of the service application-wide.

Simplified configuration: Removes the need to manually add services to the NgModule's providers array.

Lazy loading compatibility(توافق): Allows services to be scoped to specific feature modules, enhancing modularity and lazy loading.

26-What's the difference between Navigate vs navigateURL?

In Angular, navigate and navigateByUrl are two methods provided by the Router service to programmatically navigate between routes.

1- **URL Construction:**
   - **navigate**: Constructs the URL from an array of commands or segments.
   - **navigateByUrl**: Uses the URL provided as a single string or UrlTree.

2- **Use Cases:**
   - **navigate**: Preferred when building the URL from multiple parts or when dealing with relative paths.
   - **navigateByUrl**: Preferred when you have the complete URL as a string or when dealing with absolute paths.

27-What's the ChangeDetectionStrategy?

The ChangeDetectionStrategy in Angular is a configuration option that determines how Angular checks for changes and updates the view in response to those changes.

1- **(ChangeDetectionStrategy.Default):**
   - This is the default strategy used by Angular.
   - In this strategy, Angular automatically checks the entire component tree from top to bottom for any changes in component data and updates the view accordingly.
   - This strategy is suitable for most applications but can be less efficient for large applications with many components, as it can lead to frequent and potentially unnecessary checks.

2- **OnPush (ChangeDetectionStrategy.OnPush):**
   - In this strategy, Angular performs change detection only when it explicitly detects that an input property of the component has changed or an event is triggered within the component.
   - This strategy can improve performance by reducing the number of change detection cycles, making it more efficient for applications with a large number of components.

28-What's the ViewEncapsulation and how do you apply it ?

ViewEncapsulation in Angular defines how styles are applied to components and their children.
(Shadow DOM) is a web standard that provides encapsulation for DOM and CSS.

1-Emulated (Default):
Emulates (يحاكي) Shadow DOM behavior by adding unique attributes to the component's host element and its descendants, ensuring styles are scoped to the component.(`[_ngcontent-ng-c1624618894]`)

2-Shadow DOM (ViewEncapsulation.ShadowDom):
Uses the browser's native Shadow DOM to encapsulate styles, meaning styles defined in the component won't affect the rest of the application.

3-None (ViewEncapsulation.None):
No encapsulation. Styles defined in the component will affect the entire application.

29- What's the difference between pipe and directive?

Pipes:

- Pipes are used to transform data in the template. They take data as input and return transformed data as output.
- Pipes are used in template expressions, usually within interpolation or binding syntax.
- Built-in pipes include DatePipe, CurrencyPipe, UpperCasePipe, etc.
- Implemented by creating a class that implements the PipeTransform interface and decorated with the @Pipe decorator.

Directives:

- Directives are used to change the behavior or appearance of DOM elements. They can add, remove, or manipulate elements and their attributes.
- Directives are applied to elements in the template. They can be structural or attribute directives.
- Implemented by creating a class that is decorated with the @Directive decorator.

30- What is the main http module?

In Angular, the main HTTP module used for making HTTP requests is HttpClientModule. It provides a simplified API for making HTTP requests and handling responses.

31- Is Constructor lifecyclehooks?

- The constructor is not considered a lifecycle hook in Angular. The constructor is a standard TypeScript feature used to initialize class instances. While it plays a crucial role in the creation of an Angular component, it is not part of the Angular lifecycle hooks.
- Used to initialize the class and its members and Typically used for dependency injection in Angular.
- Called when the class is instantiated(إنشاء مثيل لها), which happens before Angular sets up the component or directive's input properties.

32- What is life cycle hocks ?

Lifecycle hooks in Angular are a set of methods that provide developers with the ability to tap into(الأستفاده) key moments in the lifecycle of a component or directive. These hooks allow you to perform custom actions during these moments, such as initialization, changes in input properties, and cleanup before the component or directive is destroyed.

1- **ngOnChanges**: Called before ngOnInit and whenever one or more data-bound input properties change.

2-**ngOnInit**: Called once, after the first ngOnChanges and before the component's view is rendered and Good for initializing data-bound properties(خاصية مرتبطة بالداتا) or fetching data.

3-**ngDoCheck**: Called during every change detection run, immediately after ngOnChanges and ngOnInit.

4- **ngAfterContentInit**: Called once after the first ngDoCheck and after Angular projects external content into the component's view.

5- **ngAfterContentChecked**: Called after every ngDoCheck and after Angular checks the content projected into (المسقط) the component.

6- **ngAfterViewInit**: Called once after the first ngAfterContentChecked and after Angular initializes the component's view and its children

7- **ngAfterViewChecked**: Called after every ngAfterContentChecked and after Angular checks the component's view and its children.

8- **ngOnDestroy**: Called once just before the component or directive is destroyed.

33- How angular project works? what is loaded first?

1. **Angular CLI Build Process:**
   - **TypeScript Compilation**: The Angular CLI compiles TypeScript files into JavaScript.
   - **Bundling and Optimization**: It bundles the compiled JavaScript files along with other assets (CSS, HTML, images) into a few optimized files for efficient(الفعال) loading.
2. **Loading the Application:**
   - **Index.html:** The browser first loads index.html, which contains a reference to the main JavaScript bundle generated by the Angular CLI.<app-root></app-root>
   - **Main.ts:** This is the entry point of the Angular application. It bootstraps the root module
3. **Bootstrapping:**
   - **AppModule**: The root module (AppModule) is loaded first. This module defines the main structure and configuration of the application.
   - **AppComponent**: The root component (AppComponent) associated with (مرتبط ب)AppModule is instantiated and its template is rendered.

34- What is the bootstrapping module?

The bootstrapping module in an Angular application is the root module that Angular loads to launch the application. This module is typically named AppModule, and it is responsible for configuring the main components, services, and other dependencies needed for the application to run.

35- How can we apply Error handling?

Error handling in Angular can be approached in several ways, depending on where the errors might occur and how you want to handle them. Here are some common strategies for error handling in an Angular application:

1- **HTTP Error Handling:**
   When making HTTP requests using Angular's HttpClient, you can handle errors using the catchError operator from RxJS.
   return this.http.get<any>(this.apiUrl).pipe(catchError((err:HttpErrorResponse)=>{throw err.error}

2- **Global Error Handling:** You can create a global error handler by implementing ErrorHandler and providing it in your app module.

3- **Error Handling in Components:**

```
this.dataService.getData().subscribe({
      next: (data) => this.data = data,
      error: (error) => this.errorMessage = error.message
    });
```

4- **Custom Error Pages:** You can create custom error pages for different HTTP error statuses using Angular's router.

5- **Error Interceptor :** You can create an HTTP interceptor to handle errors globally for all HTTP requests.

```
return next.handle(newreq).pipe(catchError((err:HttpErrorResponse)=>{
      console.log('dsdsd')
      throw err.error
    }))
```

36-What is the Routing Guard in Angular and its types?

Routing Guards in Angular are used to control navigation to and from different routes in an application. They allow you to add checks and handle scenarios where access to(الوصول) certain routes needs to be restricted based on conditions such as authentication, authorization, or other business logic. Angular provides several types of guards,

1. **CanActivate**: This guard is used to determine if a route can be activated.
2. **CanActivateChild**: This guard is similar to CanActivate but is specifically used to determine if a child route can be activated.
3. **CanDeactivate**: This guard is used to determine if a route can be deactivated. It is useful when you want to check if there are unsaved changes on a form and prompt the user before leaving the route.
4. **Resolve**: This guard is used to pre-fetch data before navigating to a route.
5. **CanLoad**: This guard is used to determine if a module can be loaded.

37 - What do you know about *ngTemplateOutlet Directive?

ng-template: This is a directive that defines an Angular template. It doesn't render any content by itself but can be referenced and used elsewhere in the application.

ngTemplateOutlet: This directive is used to insert the content of an ng-template into the DOM. It binds a template reference to a location in the DOM

```
<ng-template #myTemplate let-name="name">
    <div>
      Hello, {{ name }}!
    </div>
  </ng-template>

  <div *ngTemplateOutlet="myTemplate; context: { name: 'Kerolos' }"></div>
```