Json page: angular.json, package.json, tsconfig.json

npm install @angular/forms @angular/router @angular/common @angular/http

|  |  |
| --- | --- |
| Index.html | (HTML Entry Point) root component’s selector  (<app-root></app-root>). |
| Main.ts | responsible for bootstrapping the Angular application.  **bootstrapApplication(AppComponent, appConfig)**  **.catch((err) => console.error(err));** |
| app.component.ts | bootstrapApplication(AppComponent, appConfig) is called, which boots the Angular application by loading the AppComponent.  import { Component } from '@angular/core';  import { RouterOutlet } from '@angular/router';  import { RouterModule } from '@angular/router'; // Import RouterModule  @Component({    selector: 'app-root',    imports: [RouterOutlet, RouterModule], // Include RouterModule here    templateUrl: './app.component.html',    styleUrls: ['./app.component.css']  })  export class AppComponent {    title = 'AngularCRUD';  } |
| App.component.html  (Component Template) | HTML template associated with the AppComponent. It defines the view structure that will be displayed in the browser.  It will find the <router-outlet></router-outlet> directive, which is used for displaying routed views (based on the route configuration in app.routes.ts).    <nav>      <ul>        <li><a routerLink="/login" **routerLinkActive**="active">Login</a></li>        <li><a **routerLink**="/customer" **routerLinkActive**="active">Customer</a></li>        <li><a **routerLink**="/add-to-cart" **routerLinkActive**="active">Add to Cart</a></li>        <li><a **routerLink**="/payment" **routerLinkActive**="active">Payment</a></li>      </ul>    </nav>  **<router-outlet></router-outlet>** |
| app.config.ts (App Configuration) | configuration for the application, such as change detection and routing.  // app.config.ts  **import { ApplicationConfig, provideZoneChangeDetection } from '@angular/core';**  import { provideRouter } from '@angular/router';  import { routes } from './app.routes';  // Import routes here  export const appConfig: ApplicationConfig = {    providers: [provideZoneChangeDetection({ eventCoalescing: true }),      provideRouter(routes)]  }; |
| app.routes.ts (Routing Configuration) | import { Routes } from '@angular/router';  import { LoginComponent } from './login/login.component';  import { CustomerComponent } from './customer/customer.component';  import { AddToCartComponent } from './add-to-cart/add-to-cart.component';  import { PaymentComponent } from './payment/payment.component';  // Define your routes  export const routes: Routes = [    { path: '', redirectTo: 'login', pathMatch: 'full' },    { path: 'login', component: LoginComponent },    { path: 'customer', component: CustomerComponent },    { path: 'add-to-cart', component: AddToCartComponent },    { path: 'payment', component: PaymentComponent },  ]; |

**Sequence of Events When Page Loads:**

1. **Browser loads index.html**:
   * The root HTML page is loaded and rendered in the browser.
   * The Angular app's JavaScript and other resources are loaded by the browser.
2. **main.ts is executed**:
   * Angular bootstraps the application by calling bootstrapApplication(AppComponent, appConfig).
   * AppComponent is initialized, and routing configurations are set up.
3. **AppComponent is initialized**:
   * The root component (AppComponent) is created and initialized.
   * If any lifecycle hooks like ngOnInit are present, they are executed.
4. **app.component.html is rendered**:
   * Angular processes the template and renders the content, including the router outlet, navigation bar, and other static elements.
5. **Routing configuration is set up**:
   * The routes defined in app.routes.ts are available to the application.
   * When a route is activated (e.g., when a user clicks a link), Angular uses the router to display the appropriate component in the <router-outlet>.
6. **Final rendering**:
   * The application is fully initialized, and the content is rendered to the page.

|  |  |
| --- | --- |
|  |  |
| @Component({  selector: 'app-root',  template: `<**app-child [username]="parentName"></app-child**>`  })  export class AppComponent {  parentName = 'John Doe';  } | @Component({  selector: 'app-child',  template: `<p>Welcome, {{ username }}!</p>`  })  export class ChildComponent {  @Input() username!: string;  } |

Child to parentUsing @Output() and EventEmitter)

|  |  |
| --- | --- |
| Child Component (child.component.ts) | Parent Component (app.component.ts) |
| @Component({  selector: 'app-child',  template: `<button (click)="sendMessage()">Send Message</button>`  })  export class ChildComponent {  @Output() **messageEvent** = new EventEmitter<string>();  sendMessage() {  this.messageEvent.emit("Hello from Child!");  }  } | @Component({  selector: 'app-root',  template: `<app-child (**messageEvent**)="receiveMessage($event)"></app-child>  <p>Message from child: {{ message }}</p>`  })  export class AppComponent {  message = '';  receiveMessage(event: string) {  this.message = **event**;  }  } |

Sibling Component Communication (Using a Shared Service)

|  |
| --- |
| @Injectable({ providedIn: 'root' })  export class MessageService {  private messageSource = new BehaviorSubject<string>("Initial Message");  currentMessage = this.messageSource.asObservable();  updateMessage(newMessage: string) {  this.messageSource.next(newMessage);  }  } |
| @Component({  selector: 'app-sender',  template: `<input [(ngModel)]="message">  <button (click)="sendMessage()">Send</button>`  })  export class SenderComponent {  message = '';  constructor(private messageService: MessageService) {}  sendMessage() {  this.messageService.updateMessage(this.message);  }  } |
| @Component({  selector: 'app-receiver',  template: `<p>Received: {{ message }}</p>`  })  export class ReceiverComponent {  message!: string;  constructor(private messageService: MessageService) {  this.messageService.currentMessage.subscribe(msg => this.message = msg);  }  } |