Let’s set up fundamental routing for the Book Inventory page and the Contact page. We will do this instead of using both directives in one page.

1. Routes File: app.routes.ts

We will do the following:

- **Empty Path Redirect:**
  When the URL is empty (i.e. the user navigates to the root), the router redirects to `/book-inventory`.

- **Defined Routes:**
  There are two main routes:

  - `/book-inventory` loads the `BookInventoryComponent`
  - `/contact` loads the `ContactComponent`

- **Wildcard Route:**
  Any URL that does not match the above routes will be caught by the wildcard (`**`).

```
// src/app/app.routes.ts
import { Routes } from '@angular/router';
import { BookInventoryComponent } from './book-inventory/
book-inventory.component';
import { ContactComponent } from './contact/
contact.component';

export const appRoutes: Routes = [
  // Redirect the empty path to '/book-inventory'
  { path: '', redirectTo: '/book-inventory', pathMatch:
'full' },
  { path: 'book-inventory', component:
BookInventoryComponent },
  { path: 'contact', component: ContactComponent },
  // Wildcard route: any unmatched URL will redirect to '/
app'
  { path: '**', redirectTo: ‘/book-inventory', pathMatch:
'full' }
```

```
];
```

2. App Component Template: app.component.html

We will remove both directives and replace them with the following:

- **Navigation Links:**
  - Each <a> element uses the `routerLink` directive to define the target URL.
  - The `routerLinkActive="activelink"` directive automatically applies the CSS class `activelink` when the link's route is active.
  - The `[routerLinkActiveOptions]="{ exact: true }"` on the first link ensures that the active class is applied only when the URL exactly matches `/book-inventory`. (You might consider doing this on both links if exact matching is desired.) You will also have to style the class that gets added to the active links. You can use the global stylesheet.
- **Router Outlet:**
  - `<router-outlet></router-outlet>` is where Angular loads the component corresponding to the current route.:

```
<!-- src/app/app.component.html -->
<nav>
    <a routerLink="/book-inventory"
routerLinkActive="activelink" [routerLinkActiveOptions]="{
exact: true }">Book Inventory</a>
    <a routerLink="/contact"
routerLinkActive="activelink">Contact</a>
</nav>
<router-outlet></router-outlet>
```

3. Bootstrapping with Routing: main.ts

This file contains the following, with the routing information added by you:

- **Standalone Bootstrap:**
  Using Angular 17's `bootstrapApplication`, your root component (`AppComponent`) is bootstrapped as a standalone component.
- **Routing Provider:**
  `provideRouter(appRoutes)` supplies your routing configuration to the application.
- **Error Handling:**
  If there's an error during bootstrapping, it's caught and logged.

```
// src/main.ts
import { bootstrapApplication } from '@angular/platform-
browser';
import { provideRouter } from '@angular/router';
import { AppComponent } from './app/app.component';
import { appRoutes } from './app/app.routes';

bootstrapApplication(AppComponent, {
  providers: [provideRouter(appRoutes)]
}).catch(err => console.error(err));
```