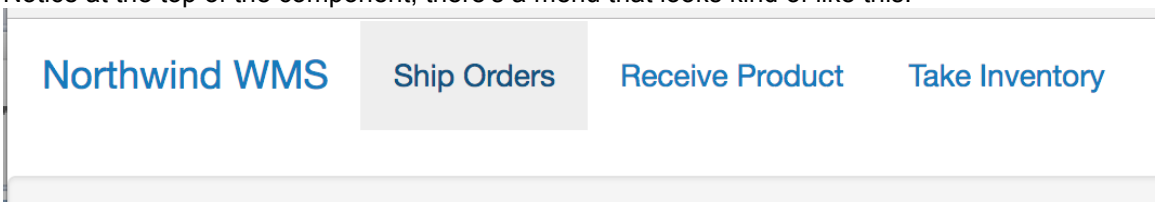
Routing Lab

So far we have some pretty cool components but you'd be hard-pressed to call it a site or a single-page app. None of the components allow us to navigate to any of the others. We'll fix that in this lab. By the time we're through, our users will be able to visit any of our pages and get to any other.

Defining the routes

1. First, make sure the AppComponent is your startup component. Go ahead and run it to make sure.
2. Notice at the top of the component, there's a menu that looks kind of like this:



3. Click on any of the three options. They don't do anything yet.

Creating the routing module

The first step would be to design the routes. In this case they're pretty obvious since so far we only have five places to go. So let's go with these routes:

Component	URL
Landing page	/
Orders to ship	/ship
Receive product	/receive
Ship a single order	/ship/<orderID>
Take inventory	/inventory

4. Create a new file called app.router.ts.

This file isn't like most of our other files. It doesn't have a class. Instead it should export an object which is created by calling the .forRoot() method of RouterModule. You'll set up the routes by passing a JavaScript array to it of the route objects, each of which have a path and a component. And how do you do that? Read on ...

5. Create an array. Here's a start:

```
const routes = [
  {path:"ship", component: OrdersToShipComponent},
  // Okay, you fill in the other three
];
```

6. Pass that array into .forRoot(). Something like this will work:

```
const routing = RouterModule.forRoot(routes);
```

7. Export the new object:

```
export routing;
```

Adding your new routing module

Since this is a bona fide module, we need to *imports* it in our main module.

8. Open app.module.ts.
9. Find the imports array in the @NgModule annotation. Add your routing module to it.

Creating a place for the pages to go

Now we've got some refactoring to do. The routing subsystem was built to dynamically place components into a host page/component. We don't have one of those. We're going to pull the dashboard functionality out of AppComponent and repurpose AppComponent to be our hosting page.

10. Create a new component called DashboardComponent with ng generate.
11. Edit app.component.html. Leave the header with the navigation bar in that file. Put everything else -- the main part of the page -- in dashboard.component.html.
12. Edit app.component.ts. Identify anything that will be needed to manage the header and footer. That stuff (if there is any) should stay. All the rest will move to dashboard.component.ts.
13. Run and test. Eliminate any compile errors. You should be seeing your header and nothing else.

Time to put them together again!

14. Edit app.component.html again. Add a tag like this:

```
<router-outlet></router-outlet>
```

15. Edit app.router.ts and change your default route to DashboardComponent instead of AppComponent.
16. Run and test. When you navigate to the root of your site, you should see AppComponent hosting your DashboardComponent.

You're probably thinking, "All that to get us back to where we started?!?" Be patient, the payoff is coming.

The payoff: Making all the routes work

17. Edit app.component.html and find the link for "Ship Orders". Make it look like this:

```
<a [routerLink]="['ship']">Ship Order</a>
```

18. Use that same pattern for "/receive", "/" , and "/inventory".
19. Run and test. You should be able to navigate to any component and you should see the navigation menu at the top of every one.

Programmatically activating a route

We have a problem that we can easily solve; in ship-order.component.ts, the order number to ship is hardcoded. It should be passed to the ShipOrderComponent when you click on an order in the DashboardComponent or OrdersToShipComponent.

20. Open list-of-orders.component.html and add a router link to each order . Have it route the user to "ship" but this time, pass the orderID to the route. This should take the user to the ShipOrderCompnent for the current order. (Hint: you will need to read _route.snapshot.params after having declared _route as an ActivatedRoute).
21. Run and test. You should be able to click on any order in ListOfOrdersComponent and have the ShipOrderComponent come up with the order details filled in.

Cool, right?