TLC:

i) In the first pass, draw rectangular boxes and Identify the immediate children
(Don't go to next level children, their children and grand-children. Go only one level.)

- TLC.component.html
    - *<header-component></header-component>*
    - *<nav-items-list></nav-items-list>*
      (list of homogeneous items, when opened its own html there will be *ngFor for *<nav-item></nav-item>*)
    - *<feed-and-status-container></feed-and-status-container>*
      (it's a wrapper of heterogeneous items, based on my highly strict and disciplined naming convention, if I open **feed-and-status-container.component.html**, I will see *<feed-component></feed-component>* and *<status-component></status-component>*, in any order. The order itself depends on **feed-and-status-container.component.css**)

- TLC.component.ts
- TLC.component.css
- TLC.module.ts
- TLC.routes.ts

Naming convention should be independent of your layout choices(position), for e.g. **LeftNavItemsList** is not recommended, instead use **NavItemsList**

ii) Now if I make a second pass with the same rules, I observe that feed-component is a list of homogeneous items, therefore the committee recommends the name should be *<feed-items-list></feed-items-list>*

iii) In the third pass, we go through the identified immediate children and see any of them depends on route changes(for e.g. if url matches ../*top/a/b*), if yes, we replace that particular child tag name with

<router-outlet></router-outlet> and you have to create a new entry in the routing table (TLC.routes.ts -> appRoutes will have a new entry)

<router-outlet name="jaggu"></router-outlet>
<router-outlet name="girishubabu"></router-outlet>

TLC (markup)

- **A**
    - A_X
    - <router-outlet name='jaggu'></router-outlet> ***// <f-tag></f-tag> // component F is placed here***
    - A_Y
        - **A_Y_Q**
            - <router-outlet name='girishu'></router-outlet>
- B
    - B_Z
    - **B_W**
        - <router-outlet name='ababu'></router-outlet>

Items in bold - all 3 of them have - one <router-outlet></router-outlet> each inside their respective .component.html's (A, A_Y_Q, B_W)

Master detail(chatslist -> list of individual items (on click) -> individual chat item), UI pattern Chat item detail

Chat List (Master) -> Chat Item (individual, looped in general)
Chat Item (interactive interface) ---- (on interaction: touch, drap, mouseover, click, dblclick) ----> Chat Item Preview (Detail)

(This is called Master Detail Pattern)

My requirement is that:

Whenever the URL hits this path or path-pattern:
localhost:8734/top/a/b

component A should change its behaviour or display state
& component B_W should change its behaviour or display state
& component A_Y_Q should change something within it (or in other words, component A_Y_Q should be designed as RESPONSIVE to URL changes / path changes / route changes)

Sub-requirement:

When top/a/b path is hit, component A's responsive sub-part should load component NNN, component B_W's responsive sub-part should load UUU &
Component A_Y_Q's responsive sub-part should load F

TLC.routes.ts

```
appRoutes = [
```

```
                {path: 'path1', component: D}, {path: 'path2', component: E}, {...}, {path:
'top/a/b', component: F }
        ];
        TLCModule.forRoot(appRoutes);


appRoutes will change now to:
appRoutes = [
                {path: 'path1', component: D},
                {path: 'path2', component: E},
                {...},
                {path: 'top/a/b', name: 'jaggu', component: NNN },
                {path: 'top/a/b', name: girishu, component: UUU },
                {path: 'top/a/b, name: ababu, component: F }
        ];
```

****

HeaderComponent.component.html:

```
<div class="fb-header-container">
        <home></home>
        <search-bar></search-bar>
        <profile></profile>
        <home></home>
        <user-profile-navigation-links></user-profile-navigation-links>
        <help></help>
        <user-profile></user-profile>

</div>
```

Best practice:

- Only in TLC use the <router-outlet>, whereas inside, the child components, use the named router-outlet(s), for e.g. <router-outlet name="jaggu"></router-outlet>
- In the overall app, there can be at most one unnamed <router-outlet>
- **router-outlet is a empty placeholder dummy tag, to be used at run time**
-

Revision Notes:

1. Router Table is generated by Angular, by somehow going through all the routes, inside the app it generates key-value pairs,
2. Key - paths, value - Component,
3. <router-outlet> is an empty placeholder, dummy tag, inside which run time dynamic content can be loaded,
4. Since there can be multiple router-outlet(s), any action which triggers a route change, should also inform inside which router-outlet that dynamic content can be loaded, For e.g.,

Final Routing Table as constructed by Angular
(What to load for a given path)

<table>
<tr><th>Key('path')</th><th>Value (Component)</th></tr>
<tr><td>'/a'</td><td>
<table>
<tr><th>Outlets(Where to load)</th><th>Component(What to load)</th></tr>
<tr><td>insideBPlaceHolder</td><td>AComponent</td></tr>
<tr><td>insideQPlaceHolder</td><td>XComponent</td></tr>
</table>
</td></tr>
<tr><td>'/a/b'</td><td>Like above, multiple outlet - component pairs can be specified</td></tr>
<tr><td>'/c'</td><td>Like above, multiple outlet - component pairs can be specified</td></tr>
<tr><td>'/c/:id'</td><td>Like above, multiple outlet - component pairs can be specified</td></tr>
</table>

Final DOM structure as constructed by Angular:
(Where to load)

```
<html>
        <body>
                <TLC>
                        <router-outlet>
                        <AComponent>
                                <BComponent>
                                <router-outlet name="insideAPlaceHolder">
                        <CComponent>
                                <router-outlet name="insideCPlaceHolder">
```

RouterLinks / programmatic route changes by Router.navigate() method call

RouterLink can specify both the parameters, what path and where (outlets)

5. An example RouterLink can be,
   <a [routerLink]="'/a/b'">
   <a [routerLink]="{path: '/a/b', where: 'insideCPlaceHolder'}">