①

Micro frontends using single-spa

* Install single-spa globally

  → npm install -g ~~single~~ create-single-spa

  → create-single-spa --version

  (v1.11.0 as of 24-6-20)

* Create container app

  → create-single-spa container --moduleType root-config

* The above command creates container app in a separate folder

②

* In our example name of the folder is 'container'
* Navigate to 'container' folder
* Run 'npm start'
* The container app is started at 'localhost:9000'

* Edit index.ejs
  → Add <link> tag to refer to Bootstrap [within <head>]
  → Add <header> tag within <body> tag
  → Add <main> tag within <body> tag below the <header> tag
  → <header> tag is a placeholder of Navbar
  → <main> tag contains a placeholder ~~of~~ for
    a. Home
    b. Users

③

④

③

* Create a ~~f~~ MF app for Navbar ~~in~~ using React

→ create-single-spa
      --framework react
      --dir navbar

* ~~A~~ new folder is created, in our case - 'navbar'. Our React app is placed here
   → Edit package.json
      - "start" script
        "webpack-dev-server
            --port 8500"

   → Now we have configured out NavBar MF app to run on
        localhost:8500

   → Run 'npm start'

④

* Ensure Navbar MF app is running on port 8500

Ensure that Container app is running on port 9000

* Add Navbar MF app to Container app.

   * Edit index.ejs (container app)
      → Add "react" and "react-dom" to shared dependencies import map

      → Point the values to CDN urls

      → Add "@dell/navbar" key to organizations import map
        Point this to "localhost:8500"

⑤

* Edit root-config.js within Container app

→ add a call to registerApplication () api

```
registerApplication ({
   name: "@dell/navbar",
   app: () =>
         System.import("@dell/
                     navbar"),
  activeWhen: () => true,
  customProps: {
    domElement: document.
      getElementById("mf-header")
  }
});
```

⑥

* Edit root.componentjs within Navbar app

→ Include code to show navigation markup using Bootstrap classes (Navbar)

→ Install @reach/router for routing functionality

~~m~~ npm install @reach/router

* Create ~~an~~ a new MF App for Home - in React

* Follow the same steps that we followed for Navbar app

⑦

* Ensure that the Home app runs on localhost:8600

* Register this app to be rendered only when the route (url) is "/"

* Render the app in the div having an id "mf-content"

* Edit root.component.js within Home app

  → Include code to display content using Bootstrap Jumbotron class.

⑧

* Create a new Angular MF App for Users

  → ng new users --routing --prefix users

  → Navigate to users folder

  → Execute

    ng add single-spa-angular

  → Update package.json, replace port 4200 to 8700

  → run -
    npm install

  → ~~start the app~~
    ~~npm run serve:single-spa:users~~

⑨

* Add users component

```
ng g c users
        --skipTests true
```

* Add user-detail component

```
ng g c user-detail
        --skipTests true
```

* Edit app.component.html, remove all markup, add <router-outlet> directive

* Edit App Module, add Empty Route Component to 'declarations' array

* Edit App Routing Module
→ Add APP_BASE_HREF to providers array
→ Add child routes to 'users' route

⑩

~~route~~

```
{path: '', component:UsersComponent},
{path: ':id', component:
        UserDetailComponent
}
{path: '**', component:
        Empty Route Component
}
```

* Start users app

```
npm run serve:single-spa:users
```

* Container App changes
- Edit index.ejs
  → Add "@dell/users" ~~to~~ key to organization import map. Point it to //localhost: 8700/main.js

→ Add zone.js script tag

```
<script src="https://unpkg.com/
  zone.js"> </script>
```

- Edit root-config.js
  → Register users app using registerApplication() api.

```
name: "@dell/users"
app: () => System.import
          ("@dell/users")
activeWhen:
   (location) => {
       return location.pathname.
          startsWith("/users");
   }
customProps: {
  domElement: document.
     getElementById("mf-content")
}
```

* Get Users Functionality
  - Users app
    - Create an user model
      id - number
      email - string
      firstName - string
      secondName - string
      avatar - string

  - Create user service

```
ng g s services/user
        --skipTests true
```

  - Methods
    1. getUsers() : Observable<User Model[]>
    2. getUser(id: number):
       Observable <UserModel>

- Service methods should issue HTTP get request to following REST API

https://reqres.in/api/users

- App Module
    - Add HttpClientModule reference to 'imports' array.

- Users component

    - Update html to show users data in a table

    - Update component code to call UserService.getUsers() method.

* User Detail functionality

    - Users Component
        - Ensure that every row of user has a link to navigate to UserDetail component. Use 'routerLink' directive to achieve this
        - Pass user Id as URL parameter
    - UserDetail Component
        - Read Id from the Url
        - Call UserService.getUser() method. Pass user Id as parameter to the method

        - Update html code to show user details.