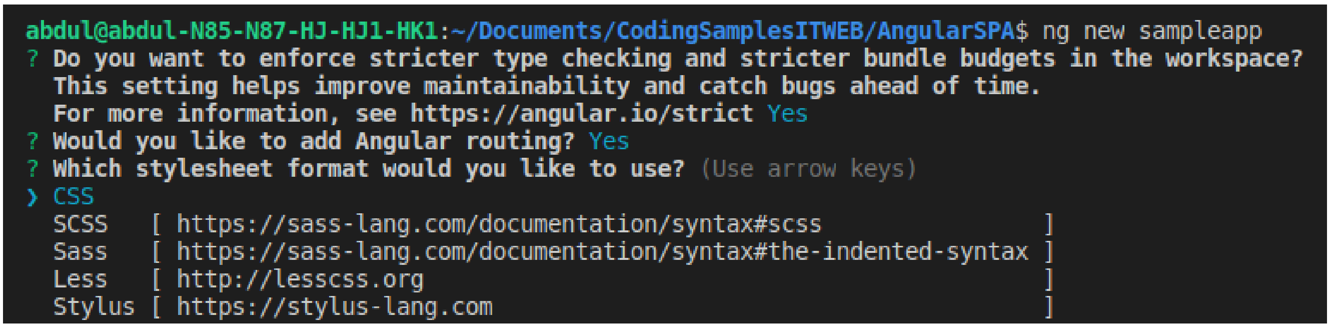
Spørgsmål 3

Question

Explain and show examples of how to design and implement a single-page application with use of Angular version 7 or newer.

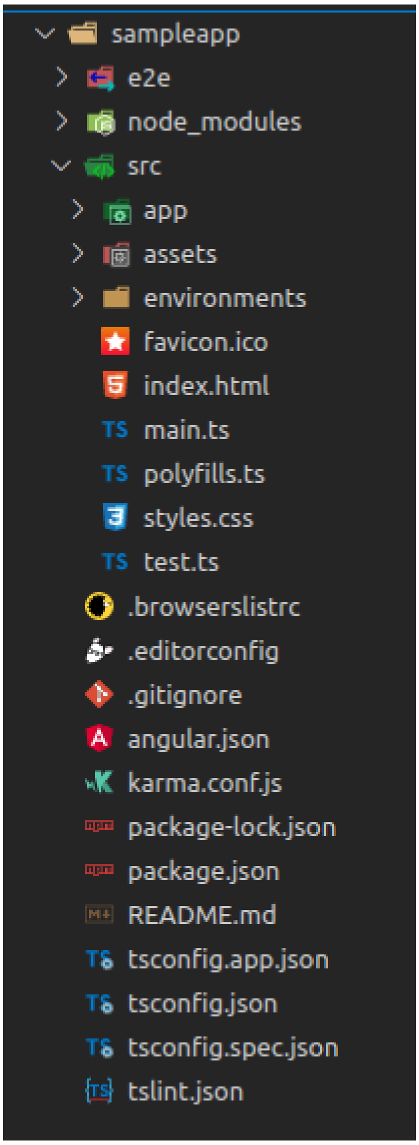
How To generate Angular boilerplate

1. npm install -g @angular/cli
2. ng new project name



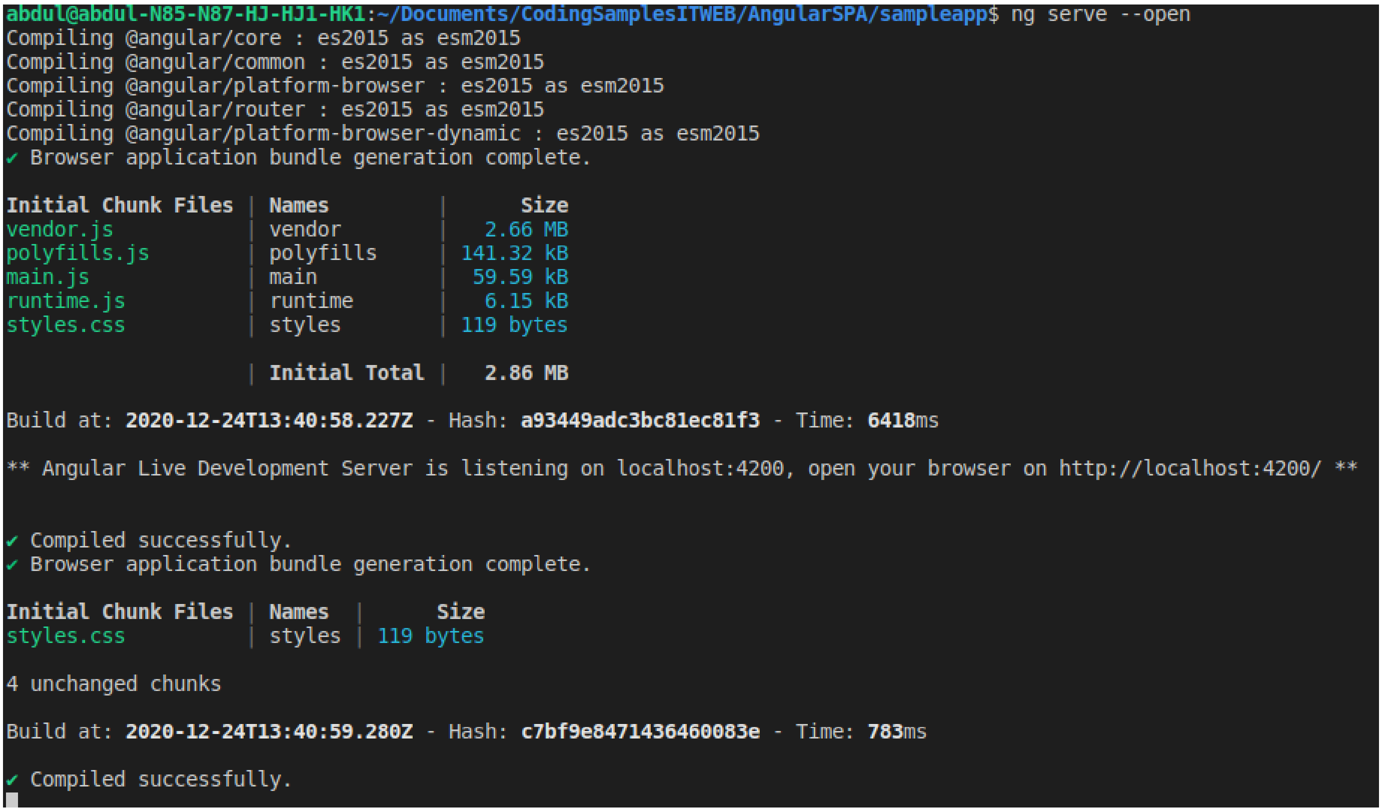
```
abdul@abdul-N85-N87-HJ-HJ1-HK1:~/Documents/CodingSamplesITWEB/AngularSPA$ ng new sampleapp
? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?
  This setting helps improve maintainability and catch bugs ahead of time.
  For more information, see https://angular.io/strict Yes
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
❯ CSS
  SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org                                             ]
  Stylus [ https://stylus-lang.com                                        ]
```

3.



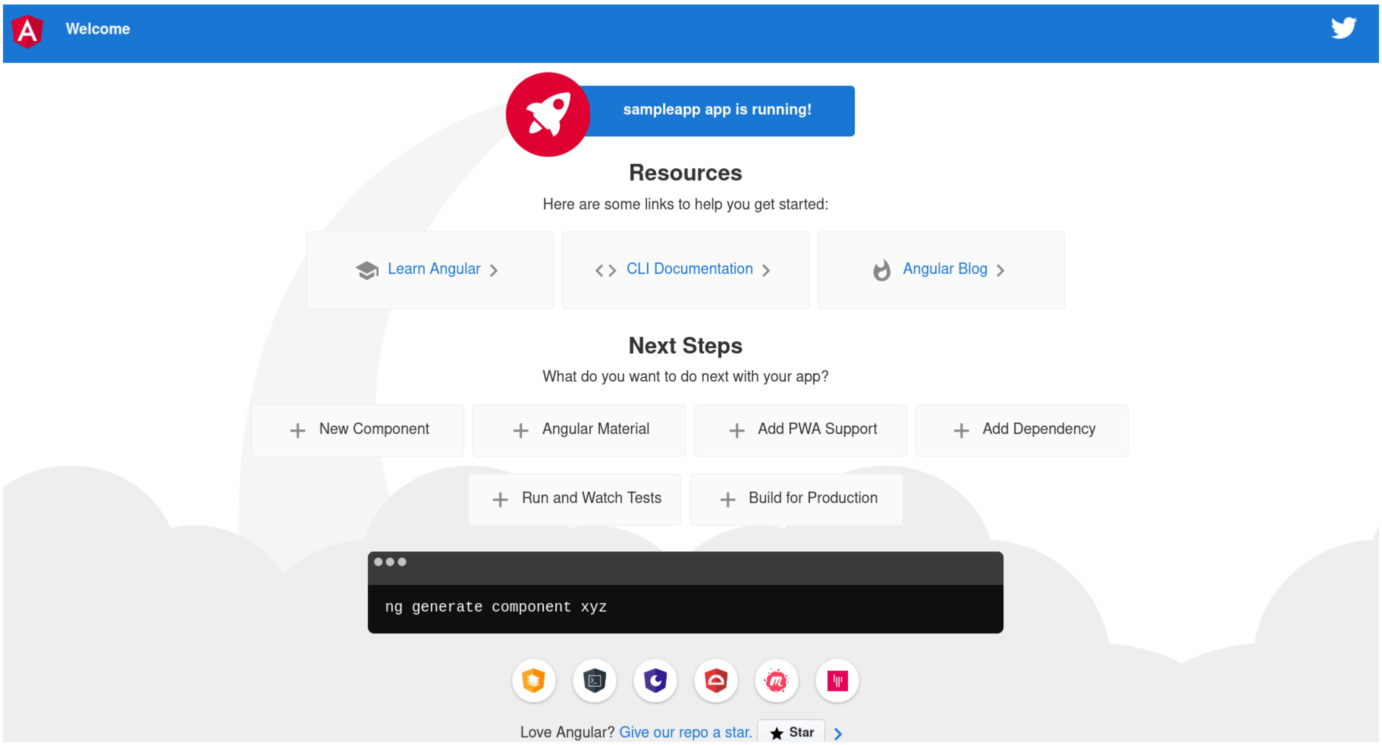
```
sampleapp
  e2e
  node_modules
  src
    app
    assets
    environments
    favicon.ico
    index.html
    main.ts
    polyfills.ts
    styles.css
    test.ts
  .browserslistrc
  .editorconfig
  .gitignore
  angular.json
  karma.conf.js
  package-lock.json
  package.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.spec.json
  tslint.json
```

4. ng serve --open



```
abdul@abdul-N85-N87-HJ-HJ1-HK1:~/Documents/CodingSamplesITWEB/AngularSPA/sampleapp$ ng serve --open
Compiling @angular/core : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/router : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
✔ Browser application bundle generation complete.

Initial Chunk Files | Names         |      Size
vendor.js           | vendor        |   2.66 MB
polyfills.js        | polyfills     | 141.32 kB
main.js             | main          |  59.59 kB
runtime.js          | runtime       |   6.15 kB
styles.css          | styles        | 119 bytes

                    | Initial Total |   2.86 MB

Build at: 2020-12-24T13:40:58.227Z - Hash: a93449adc3bc81ec81f3 - Time: 6418ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


✔ Compiled successfully.
✔ Browser application bundle generation complete.

Initial Chunk Files | Names  |      Size
styles.css          | styles | 119 bytes

4 unchanged chunks

Build at: 2020-12-24T13:40:59.280Z - Hash: c7bf9e8471436460083e - Time: 783ms

✔ Compiled successfully.
```

5. boilerplate setup complete



Resources

Here are some links to help you get started:

Next Steps

What do you want to do next with your app?

```
ng generate component xyz
```

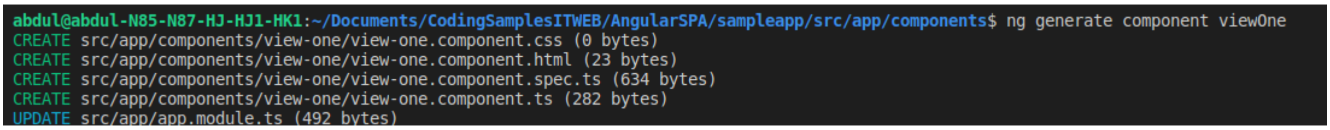
Love Angular? Give our repo a star.

How to implement singlepage capability

- TO make a SPA a routing mechanism is needed. The routing switches out the views within the depicted container on the page, without changing anything outside the container. This eliminate the need to reload the page and gives a much faster site, and much better UX.
- Angular is a framework focused around the SPA principles and therefore designed to be used when making SPA.

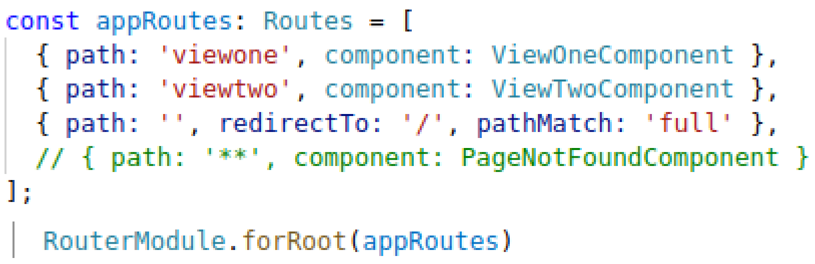
Implementation

1. Create module "ng generate component somename"



```
abdul@abdul-N85-N87-HJ-HJ1-HK1:~/Documents/CodingSamplesITWEB/AngularSPA/sampleapp/src/app/components$ ng generate component viewOne
CREATE src/app/components/view-one/view-one.component.css (0 bytes)
CREATE src/app/components/view-one/view-one.component.html (23 bytes)
CREATE src/app/components/view-one/view-one.component.spec.ts (634 bytes)
CREATE src/app/components/view-one/view-one.component.ts (282 bytes)
UPDATE src/app/app.module.ts (492 bytes)
```

2. add routing to components and import them

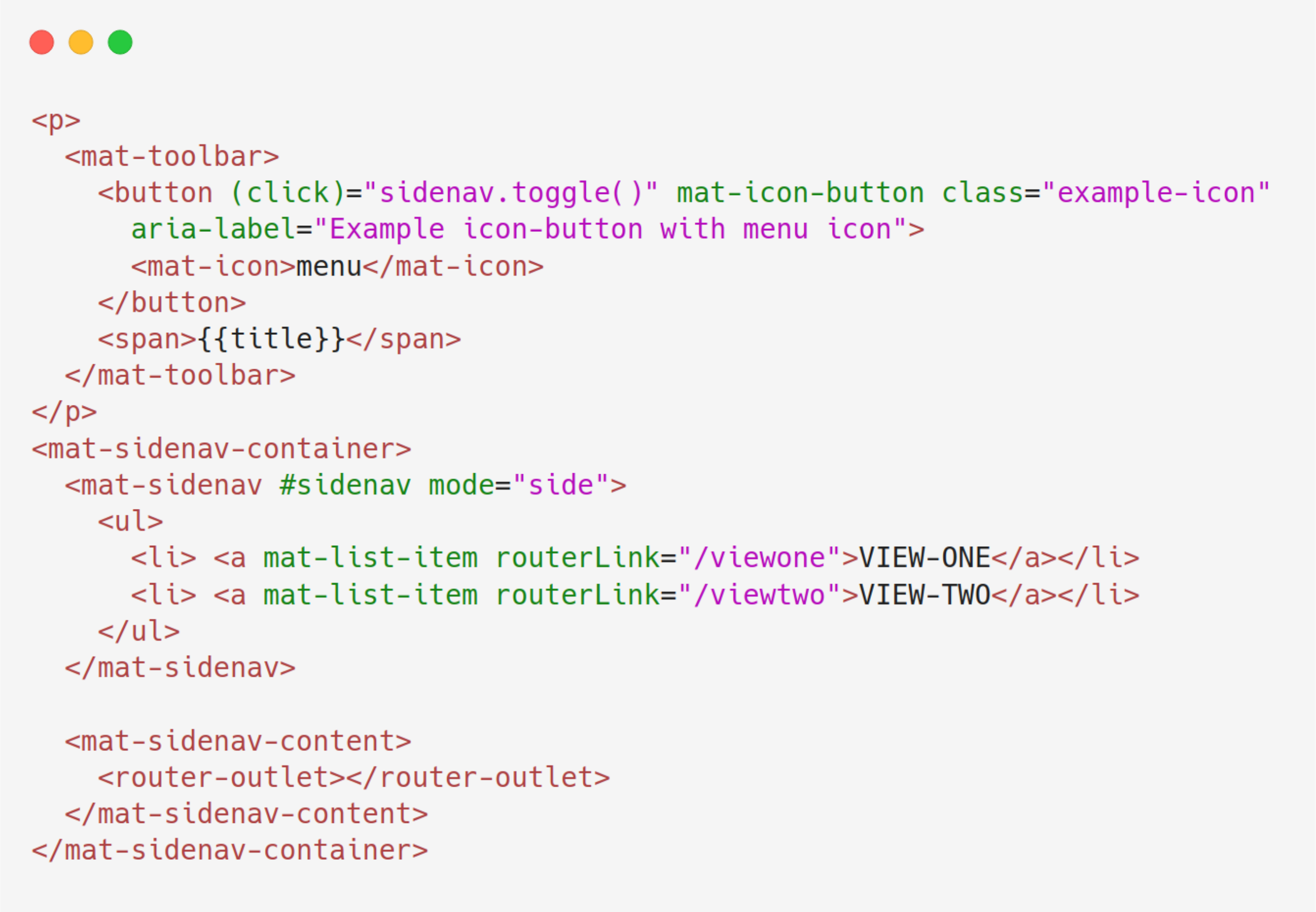


```
const appRoutes: Routes = [
  { path: 'viewone', component: ViewOneComponent },
  { path: 'viewtwo', component: ViewTwoComponent },
  { path: '', redirectTo: '/', pathMatch: 'full' },
  // { path: '**', component: PageNotFoundComponent }
];
```

```
  RouterModule.forRoot(appRoutes)
```

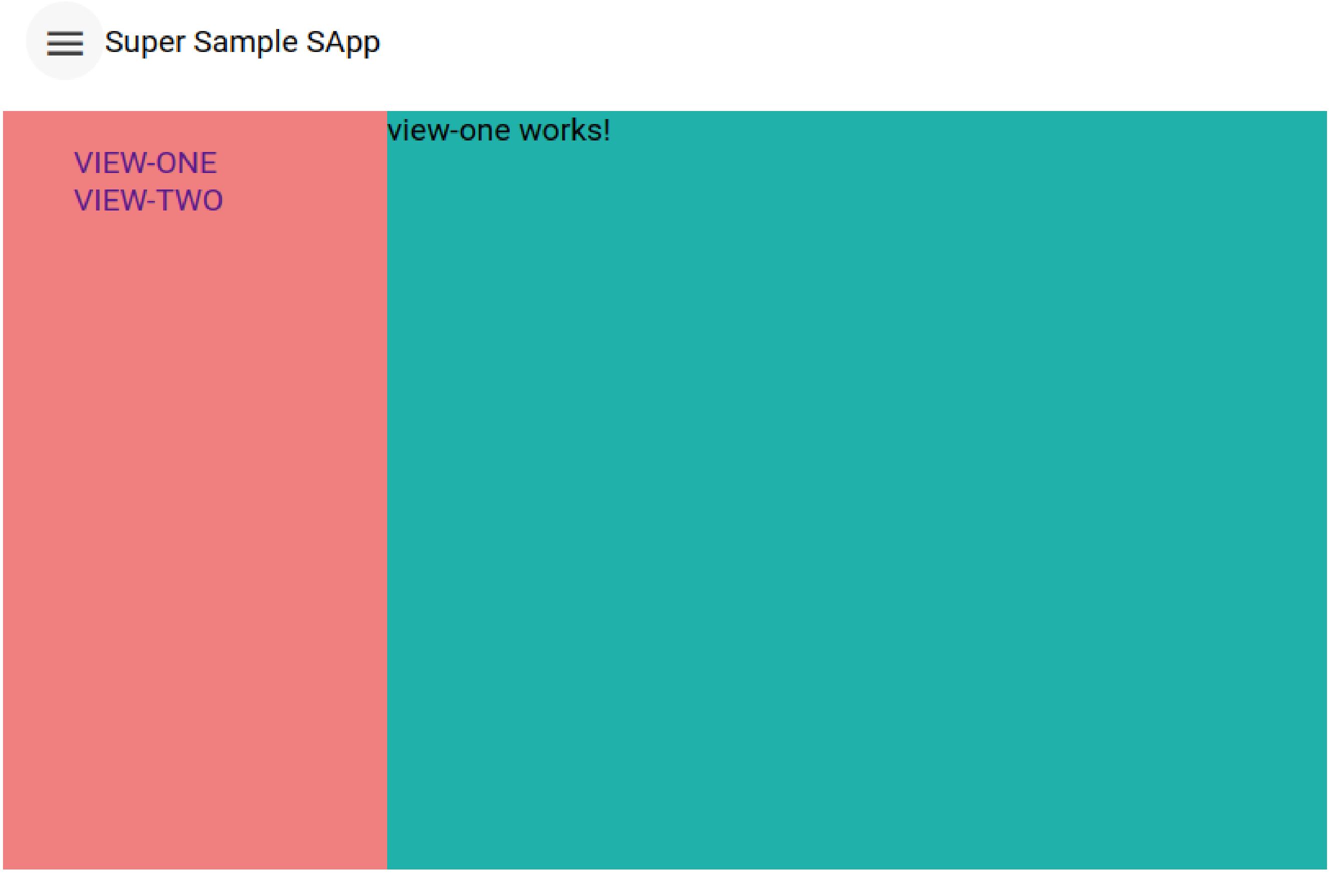
3. make sure "<router-outlet></router-outlet>" is inserted in the base page
4. Make sure "**routerLink**" attribute is used when refering to a component. This attribute is the key to switching pages in the router without reloading the whole appication. Hard reload would happen if "href" was used instead

5. Picture

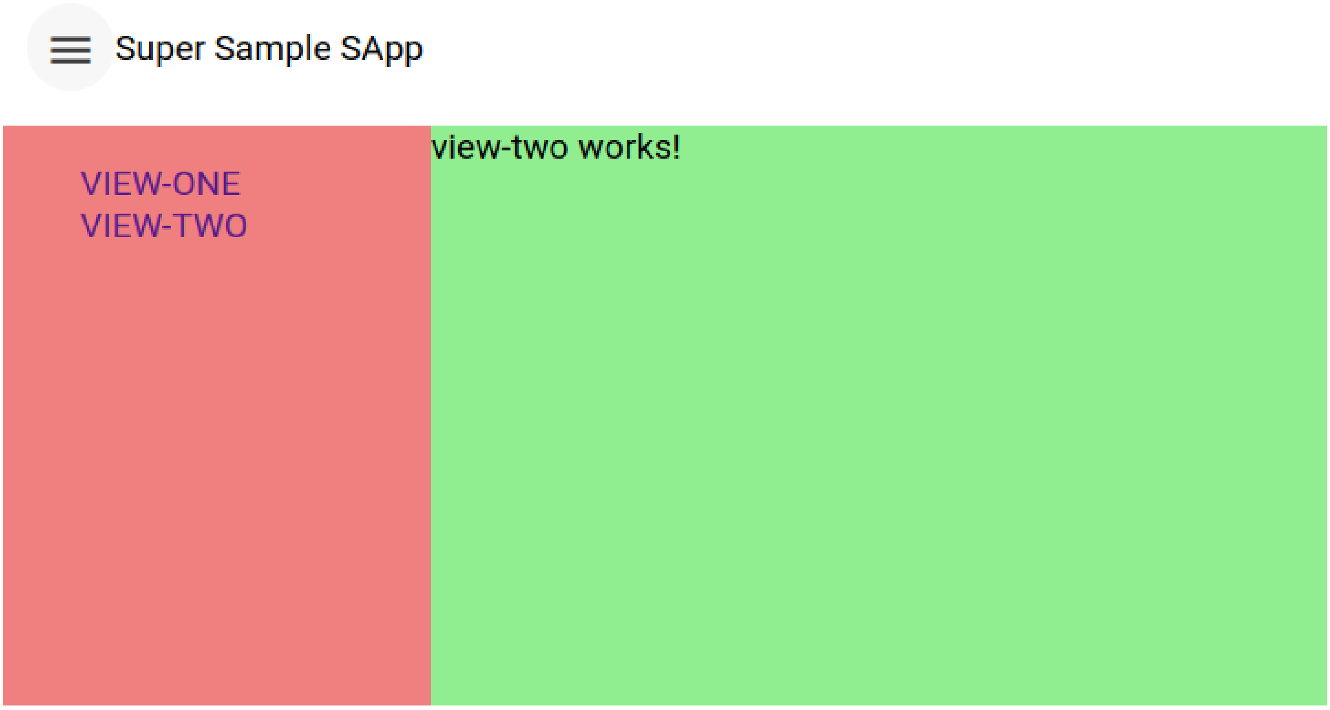


```
<p>
  <mat-toolbar>
    <button (click)="sidenav.toggle()" mat-icon-button class="example-icon"
      aria-label="Example icon-button with menu icon">
      <mat-icon>menu</mat-icon>
    </button>
    <span>{{title}}</span>
  </mat-toolbar>
</p>
<mat-sidenav-container>
  <mat-sidenav #sidenav mode="side">
    <ul>
      <li> <a mat-list-item routerLink="/viewone">VIEW-ONE</a></li>
      <li> <a mat-list-item routerLink="/viewtwo">VIEW-TWO</a></li>
    </ul>
  </mat-sidenav>

  <mat-sidenav-content>
    <router-outlet></router-outlet>
  </mat-sidenav-content>
</mat-sidenav-container>
```

6. REsults



view-one works!



view-two works!

Routing

First of all, Angular Router takes care of the duties of a JavaScript router:

- it activates all required Angular components to compose a page when a user navigates to a certain URL
- it lets users navigate from one page to another without page reload
- it updates the browser’s history so the user can use the back and forward buttons when navigating back and forth between pages.

In addition, Angular Router allows us to:

- redirect a URL to another URL
- resolve data before a page is displayed
- run scripts when a page is activated or deactivated
- lazy load parts of our application.
- subscribing. you got to sub to an observable
- Hooks
- Lifecycle