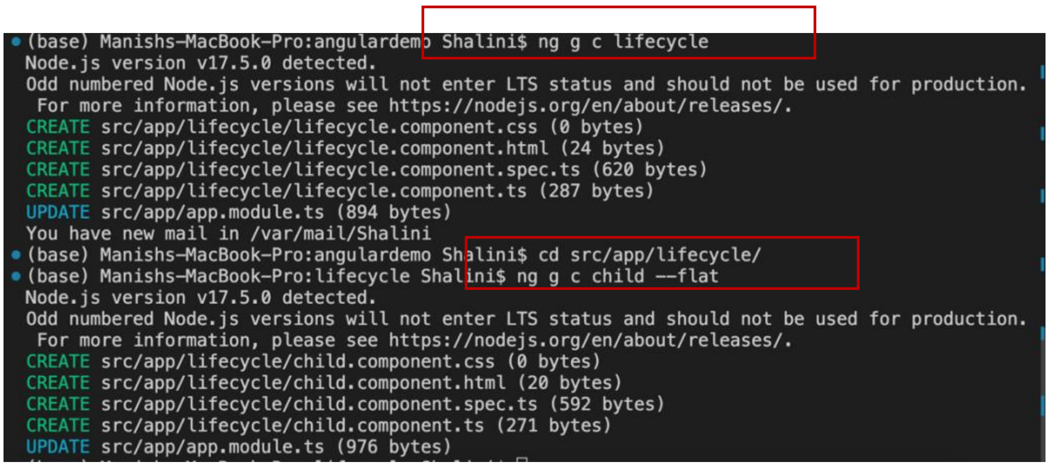
STEP 8: Lifecycle

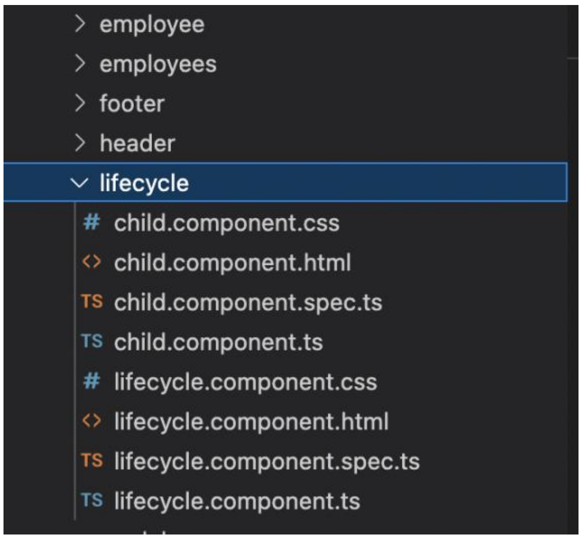
1. Add Lifecycle component by executing the below command from the angulardemo folder:
   ng g c lifecycle

2. Add child component by executing the below command from within the **lifecycle** folder as follows:
   ng g c child --flat



```
• (base) Manishs-MacBook-Pro:angulardemo Shalini$ ng g c lifecycle
  Node.js version v17.5.0 detected.
  Odd numbered Node.js versions will not enter LTS status and should not be used for production.
   For more information, please see https://nodejs.org/en/about/releases/.
  CREATE src/app/lifecycle/lifecycle.component.css (0 bytes)
  CREATE src/app/lifecycle/lifecycle.component.html (24 bytes)
  CREATE src/app/lifecycle/lifecycle.component.spec.ts (620 bytes)
  CREATE src/app/lifecycle/lifecycle.component.ts (287 bytes)
  UPDATE src/app/app.module.ts (894 bytes)
  You have new mail in /var/mail/Shalini
• (base) Manishs-MacBook-Pro:angulardemo Shalini$ cd src/app/lifecycle/
• (base) Manishs-MacBook-Pro:lifecycle Shalini$ ng g c child --flat
  Node.js version v17.5.0 detected.
  Odd numbered Node.js versions will not enter LTS status and should not be used for production.
   For more information, please see https://nodejs.org/en/about/releases/.
  CREATE src/app/lifecycle/child.component.css (0 bytes)
  CREATE src/app/lifecycle/child.component.html (20 bytes)
  CREATE src/app/lifecycle/child.component.spec.ts (592 bytes)
  CREATE src/app/lifecycle/child.component.ts (271 bytes)
  UPDATE src/app/app.module.ts (976 bytes)
```

3. Folder looks as follows:



```
> employee
> employees
> footer
> header
v lifecycle
    child.component.css
    child.component.html
    child.component.spec.ts
    child.component.ts
    lifecycle.component.css
    lifecycle.component.html
    lifecycle.component.spec.ts
    lifecycle.component.ts
```

4. Comment out the previous tags from app.component.html file and add <app-lifecylce> tag

5. Add the following in the lifecycle.component.ts

   pcountry:string ='usa';
    emp = {"name":"Shalini"}
    constructor() {

```
  console.log(`parent constructor`);
 }

 ngOnInit(): void {
  console.log(`parent ng oninit `);
 }
```

6. Add following in lifecycle.component.html

```
<div >
  <h1>parent component! {{pcountry}}</h1>
  <select [(ngModel)]="pcountry" class="form-select mb-5">
    <option value="india">India</option>
    <option value="usa">USA</option>
    <option value="uk">UK</option>
    <option value="ireland">Ireland</option>
  </select>
  <h4>{{emp | json}}</h4>
  <p><input type="text" [(ngModel)]="emp.name" placeholder="Enter
employee name"/></p>
   <p>Show Child : <input type="checkbox" [(ngModel)]="show"/></p>
  <div *ngIf="show">
  <app-child [country]="pcountry" [employee]="emp"></app-child>
  </div></div>
```

7. To pass data from parent to child component use @Input decorator.
   Add below lines in child.component.ts file

```
@Input()
country:string ='uk';
@Input()
employee = {"name":""}
```

8. Initialize some dummy data in child.component.ts

```
data:any[] =[
  {"country":"uk","states":["London"]},
  {"country":"india","states":["maharashtra", "UP","MP"]},
  {"country":"ireland","states":["ire1","ire2"]},
  {"country":"usa","states":["Illinois","SFO"]}
 ]
 countrystates:any[]=[]
```

9. Implement the OnInit, OnChanges , DoCheck interface in child.component.ts:
   export class ChildComponent implements OnInit, OnChanges , DoCheck, OnDestroy

10. Override the respective lifecycle methods in child.component.ts

```
ngDoCheck(): void {
    console.log(`child ng do check ${this.country} : ${this.employee.name}`);
  }
  constructor() {
    console.log(`child constructor ${this.country}`);
  }

 ngOnChanges(changes: SimpleChanges): void {
    console.log(`child ng on changes ${this.country} : ${this.employee.name}`);
    this.countrystates = this.data.filter(item=>item.country === this.country)

  }
  ngOnInit(): void {
    console.log(`child ng on init ${this.country}`);
  }
ngOnDestroy(): void {
    console.log(`child destroy`);
  }
```

11. Update in child.component.html with below code:
```
<div style="border: 1px solid; padding:20px;">
<h1>child component!</h1>
<h3>Country : {{country | uppercase}}</h3>
<h4 *ngFor="let state of countrystates">{{state.states }}
</h4>
<h3>{{employee | json}}</h3>
</div>
```

12. OnInit and destroy are lifecycle methods that are invoked only once the lifecyle of the component
13. Changing the country in parent will invoke ngOnchanges and ngDoCheck methods
14. Changing the employee name in parent will only invoke ngDoCheck method