Use of DB.JSON mock DB with Service

Step 1: ng g m crud/crud-module

Step 2: ng g c crud/crud-module/employee --skip-tests

Step 3: ng g c crud/crud-module/employee-list –skip-tests
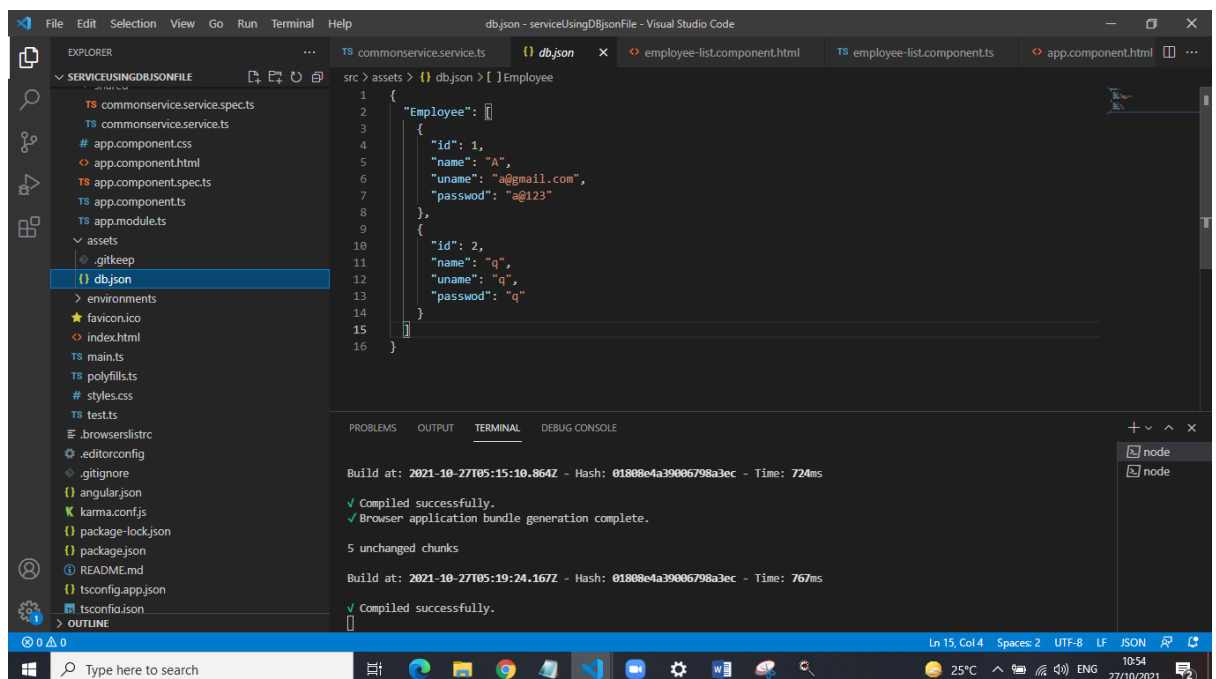
Step 4: ng g class model/Employee

Step 5: ng g s shared/common

Step 6: we have to create a dummy database for mock services, with the help of db.json file

So, right click on assets folder in src folder structure, and create one file named with db.json.

Create one Employee json in that file as mentioned below.

Step 7: To activate and communicate with db.json file, we have one server called, json-server.

So, we have to install that server in project, with help of following command.

npm i json-server –g

Step 8: Load the db.json file on json server

➢ Go to the assets folder and right click on the folder > open in integrated terminal
➢ The on terminal, it will give a path till assets folder
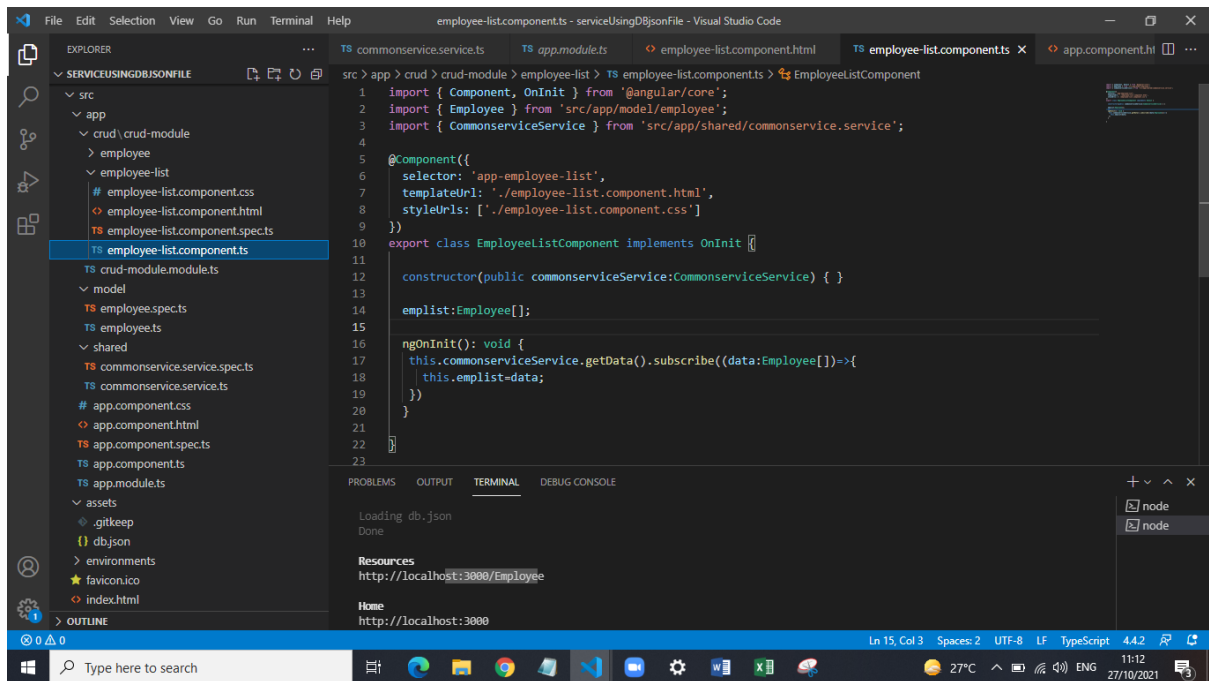➢ Enter a command

- o Json-server --watch db.json

Step 9: Register the HttpClientModule in parent module
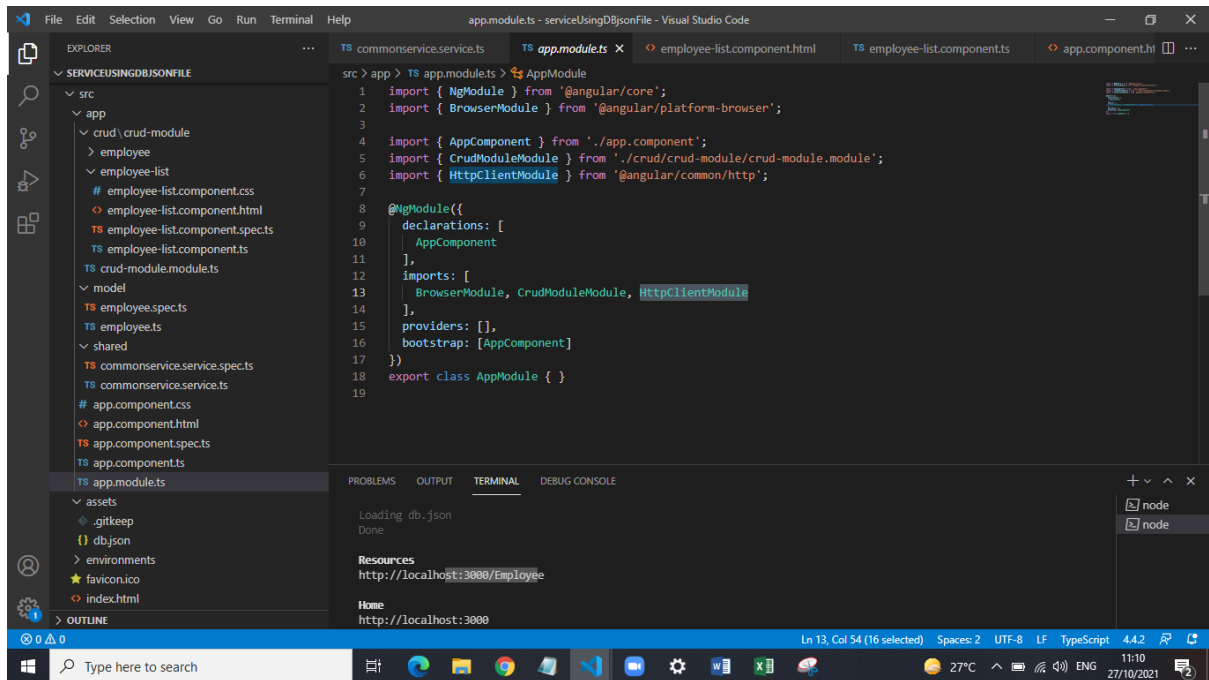
Step 10: Create object of HttpClient in Common Service
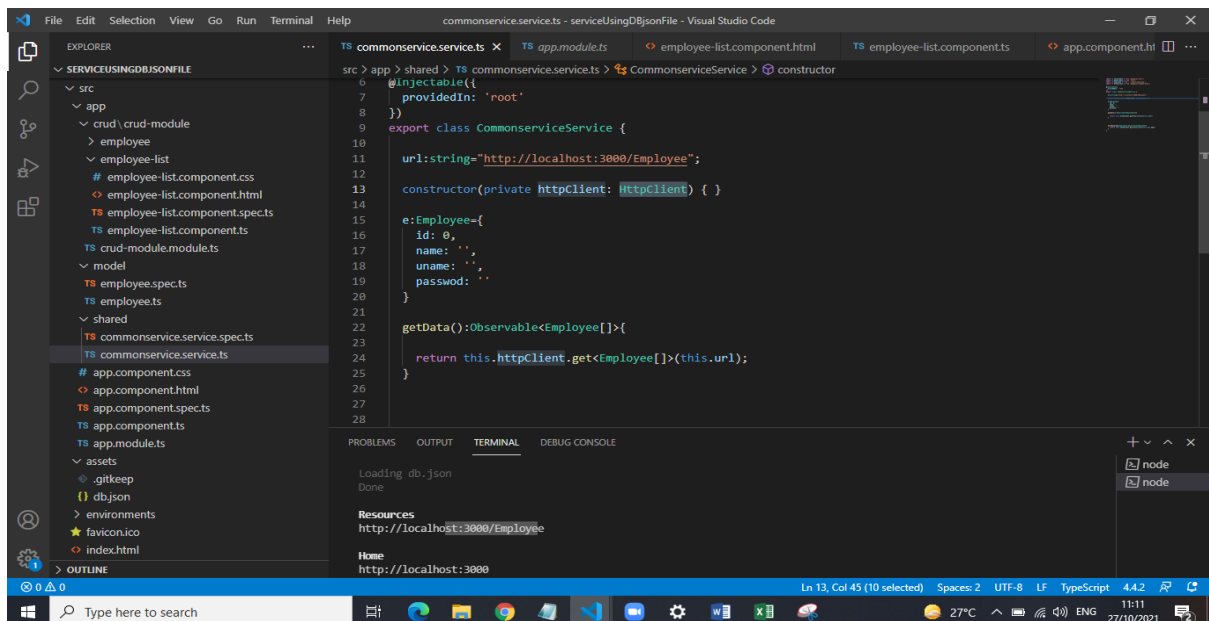
Step 11:


```typescript
import { Component, OnInit } from '@angular/core';
import { Employee } from 'src/app/model/employee';
import { CommonserviceService } from 'src/app/shared/commonservice.service';

@Component({
  selector: 'app-employee-list',
  templateUrl: './employee-list.component.html',
  styleUrls: ['./employee-list.component.css']
})
export class EmployeeListComponent implements OnInit {

  constructor(public commonserviceService:CommonserviceService) { }

  emplist:Employee[];

  ngOnInit(): void {
    this.commonserviceService.getData().subscribe((data:Employee[])=>{
      this.emplist=data;
    })
  }

}
```
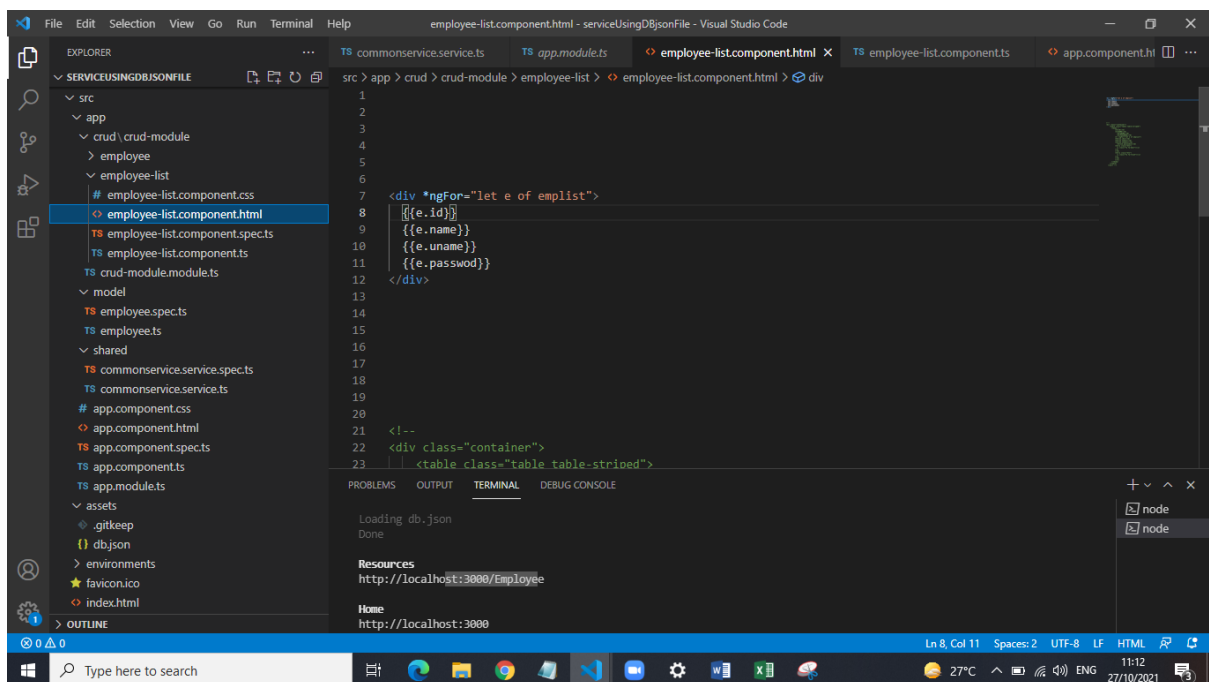
Step 12:


```html
<div *ngFor="let e of emplist">
  {{e.id}}
  {{e.name}}
  {{e.uname}}
  {{e.passwod}}
</div>

<!--
<div class="container">
    <table class="table table-striped">
```