Angular 5 CRUD Operation Using Node, Express, MongoDB

we’ll learn to create basic CRUD application using Angular 5, Nodejs, Express and MongoDB NoSQL database.

**Introduction**

We will create a demo project using Angular CLI for front-end, Node.js and Express for middle-end, and MongoDB for the back-end. In this article, we start from the beginning.

**Requirement**

* Visual Studio Code or any IDM for development.
* Node.js - if you already installed node, then check with *node –v* command and also check the npm command.
* MongoDB - In this project, we are using MongoDB database; you can also use SQL or other database.

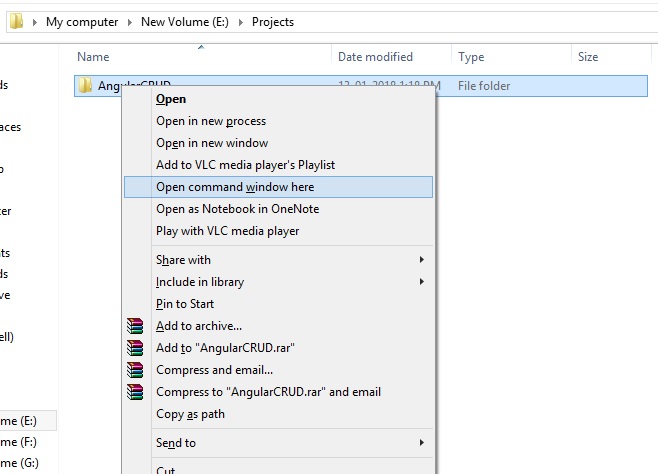
If you already installed Angular CLI, globally check the version with this command *ng -v*

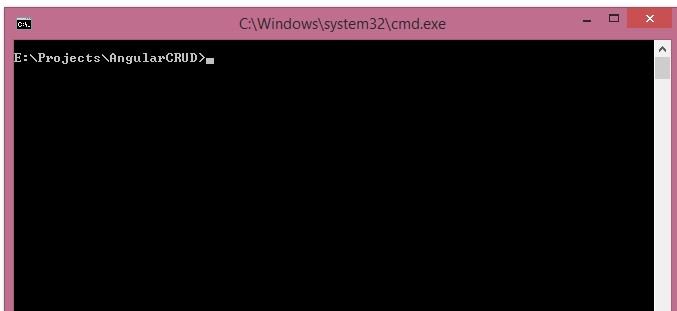
**Download links**

* Node - https://nodejs.org/en/download/
* VS code - https://code.visualstudio.com/
* MongoDB https://www.mongodb.com/download-center

Let’s start and create demo application.

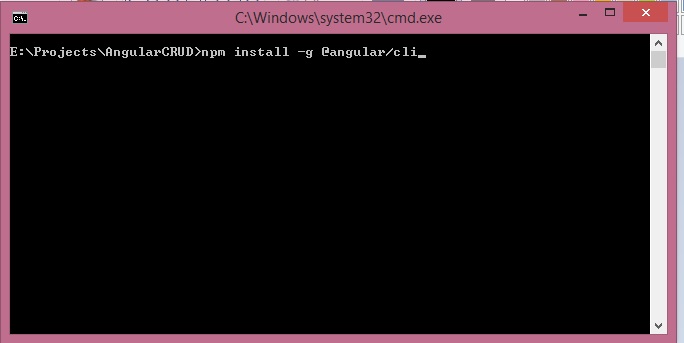
**Step 1**  
Create a new folder with any name, let's say, AngularCRUD. After the folder is created, then press ctrl+shift Right click for opening the command window here.





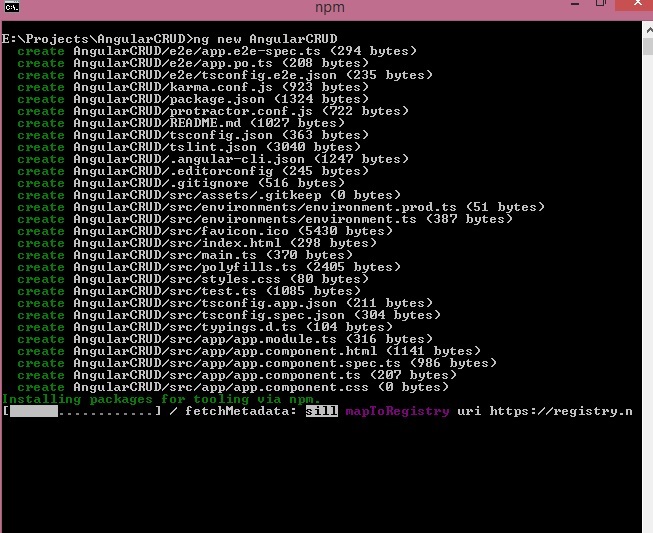
**Step 2**  
After the folder opens in the command prompt, run this command for Angular CLI to install in our folder.

*npm install -g @angular/cli*



**Step 3**  
  
If installed successfully, then run this command for creating a new application. Let's again set the project name as AngularCRUD.

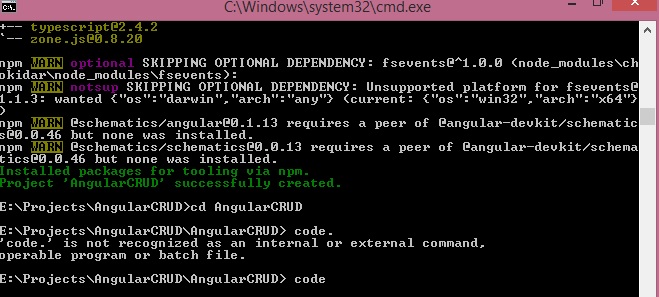
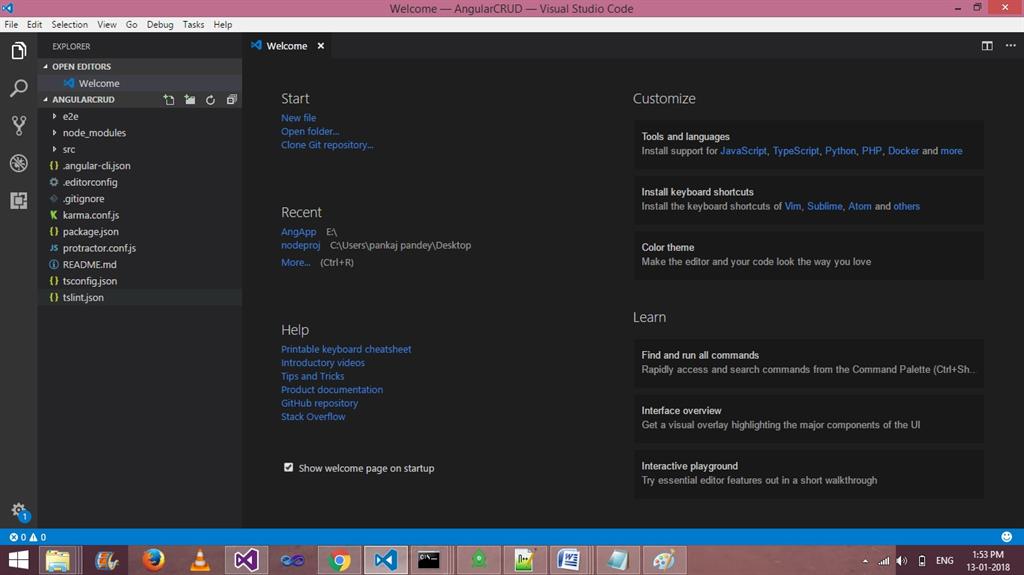
*ng new projectname*



**Step 4**   
  
When the *ng new*command is created and installed successfully, change your directory

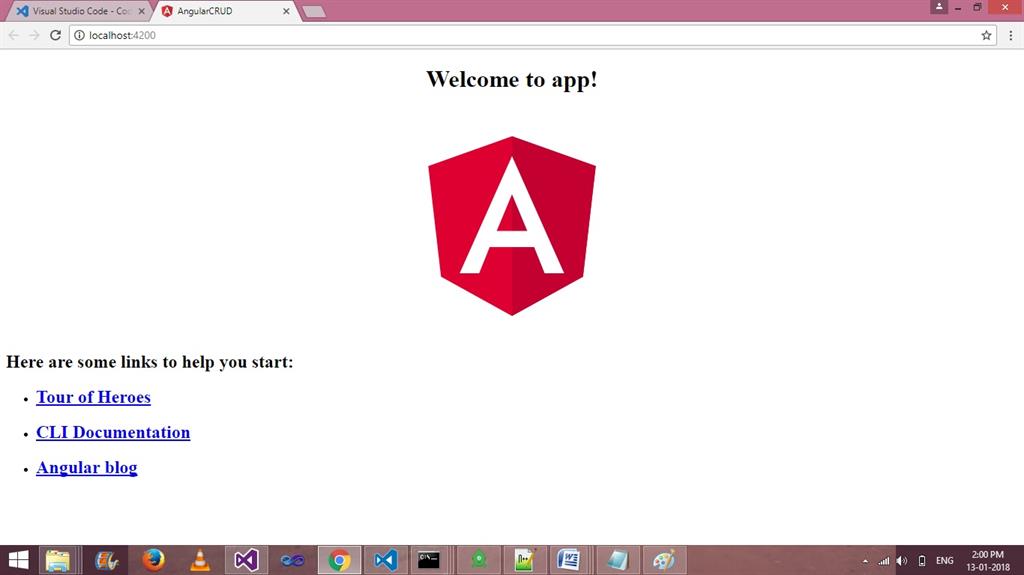
cd AngularCRUD.

**Step 5**   
  
Now, we open our project in Visual Studio code with code command like this.

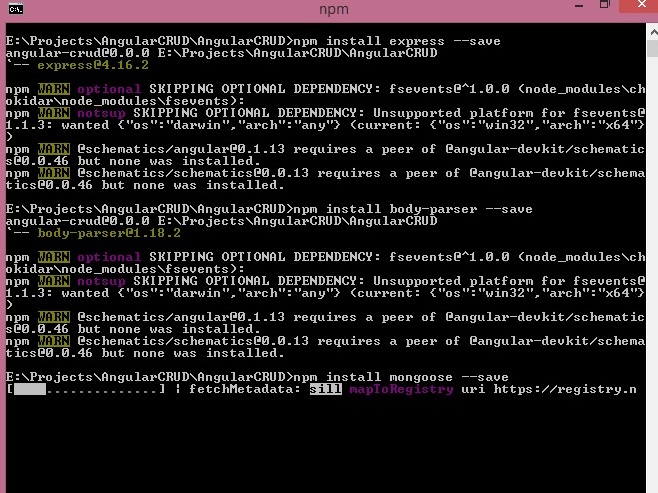
   
  
Now, we can see VSCode opened automatically.  
  
  
  
**Step 6**   
  
Now run your application by using this command -

ng serve - o

Here, -o stands for opening application in default browser.

  
 **Step 7**  
Now, let us install Express and Mongoose body parser using this command.

* npm install express --save
* npm install mongoose -- save
* npm install body-parser --save



**Step 8**After installing the above three packages, create a new file, server.js.

1. var express = require('express');
2. var path = require("path");
3. var bodyParser = require('body-parser');
4. var mongo = require("mongoose");
6. var db = mongo.connect("mongodb://localhost:27017/AngularCRUD", function(err, response){
7. **if**(err){ console.log( err); }
8. **else**{ console.log('Connected to ' + db, ' + ', response); }
9. });

12. var app = express()
13. app.use(bodyParser());
14. app.use(bodyParser.json({limit:'5mb'}));
15. app.use(bodyParser.urlencoded({extended:**true**}));

18. app.use(function (req, res, next) {
19. res.setHeader('Access-Control-Allow-Origin', 'http://localhost:4200');
20. res.setHeader('Access-Control-Allow-Methods', 'GET, POST, OPTIONS, PUT, PATCH, DELETE');
21. res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
22. res.setHeader('Access-Control-Allow-Credentials', **true**);
23. next();
24. });
26. var Schema = mongo.Schema;
28. var UsersSchema = **new** Schema({
29. name: { type: String   },
30. address: { type: String   },
31. },{ versionKey: **false** });

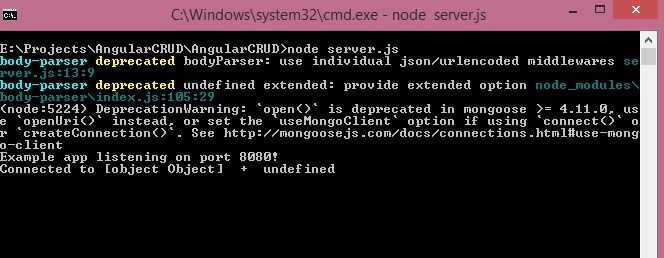
34. var model = mongo.model('users', UsersSchema, 'users');
36. app.post("/api/SaveUser",function(req,res){
37. var mod = **new** model(req.body);
38. **if**(req.body.mode =="Save")
39. {
40. mod.save(function(err,data){
41. **if**(err){
42. res.send(err);
43. }
44. **else**{
45. res.send({data:"Record has been Inserted..!!"});
46. }
47. });
48. }
49. **else**
50. {
51. model.findByIdAndUpdate(req.body.id, { name: req.body.name, address: req.body.address},
52. function(err,data) {
53. **if** (err) {
54. res.send(err);
55. }
56. **else**{
57. res.send({data:"Record has been Updated..!!"});
58. }
59. });

62. }
63. })
65. app.post("/api/deleteUser",function(req,res){
66. model.remove({ \_id: req.body.id }, function(err) {
67. **if**(err){
68. res.send(err);
69. }
70. **else**{
71. res.send({data:"Record has been Deleted..!!"});
72. }
73. });
74. })


78. app.get("/api/getUser",function(req,res){
79. model.find({},function(err,data){
80. **if**(err){
81. res.send(err);
82. }
83. **else**{
84. res.send(data);
85. }
86. });
87. })

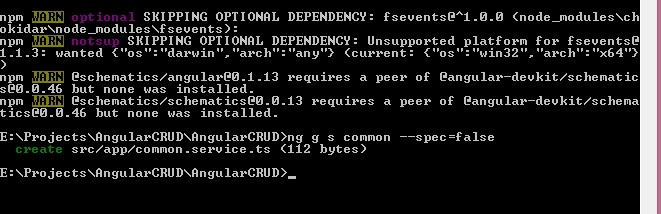
90. app.listen(8080, function () {
92. console.log('Example app listening on port 8080!')
93. })

**Step 9**   
  
Let us open the project folder in other command prompt and run the node server.js on port 8080.



**Setp 10**   
  
Create a new Angular Service for common AJAX API calling. Use thie commond

ng g s common –spec=false

   
  
**Step 11**   
  
Write the following code in common.service.ts for API.

1. import { Injectable } from '@angular/core';
2. import {Http,Response, Headers, RequestOptions } from '@angular/http';
4. import { Observable } from 'rxjs/Observable';
5. import 'rxjs/add/operator/map';
6. import 'rxjs/add/operator/do';
8. @Injectable()
9. export **class** CommonService {
11. constructor(**private** http: Http) { }
13. saveUser(user){
14. **return** **this**.http.post('http://localhost:8080/api/SaveUser/', user)
15. .map((response: Response) =>response.json())
16. }
18. GetUser(){
19. **return** **this**.http.get('http://localhost:8080/api/getUser/')
20. .map((response: Response) => response.json())
21. }
22. deleteUser(id){
23. **return** **this**.http.post('http://localhost:8080/api/deleteUser/',{'id': id})
24. .map((response: Response) =>response.json())
25. }
27. }

**Step 12**   
  
Now, write the View code in app.module.ts file.

1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
4. import { HttpModule } from '@angular/http';
5. import { FormsModule } from '@angular/forms';
7. import { AppComponent } from './app.component';
9. import {CommonService} from './common.service';

12. @NgModule({
13. declarations: [
14. AppComponent
15. ],
16. imports: [
17. BrowserModule,HttpModule,FormsModule,
18. ],
19. providers: [CommonService],
20. bootstrap: [AppComponent]
21. })
22. export **class** AppModule { }

**Step 13**

Code for app.component.html.

1. <form #userForm="ngForm"   (ngSubmit)="onSave(userForm.value)" novalidate>
2. <p>Is "myForm" valid? {{userForm.valid}}</p>
3. <table border='1'>
4. <tr>
5. <td>name</td>
6. <td>
7. <input name="id" type="hidden"     [(ngModel)]="id" />
8. <input name="name" type="text"  required  [(ngModel)]="name" />
10. </td>
11. </tr>
13. <tr>
14. <td>address</td>
15. <td>    <input name="address" required  type="text"   [(ngModel)]="address" /></td>
16. </tr>
17. <tr>
18. <td colspan="2">
19. <input type="submit" value="{{valbutton}}" />
20. </td>
21. </tr>
22. </table>
23. </form>
25. <table border='1'>
27. <tr>
28. <td>Id</td>
29. <td>Name</td>
30. <td>Address</td>
31. <td>Edit</td>
32. <td>Delete</td>
33. </tr>
34. <tr \*ngFor="let kk of Repdata;let ind = index">
36. <td>{{ind + 1}}</td>
37. <td>{{kk.name}}</td>
38. <td>{{kk.address}}</td>
39. <td><a (click)="edit(kk)" style="color:blueviolet">Edit</a></td>
40. <td><a (click)="delete(kk.\_id)" style="color:blueviolet">Delete</a>  </td>
41. </tr>
42. </table>

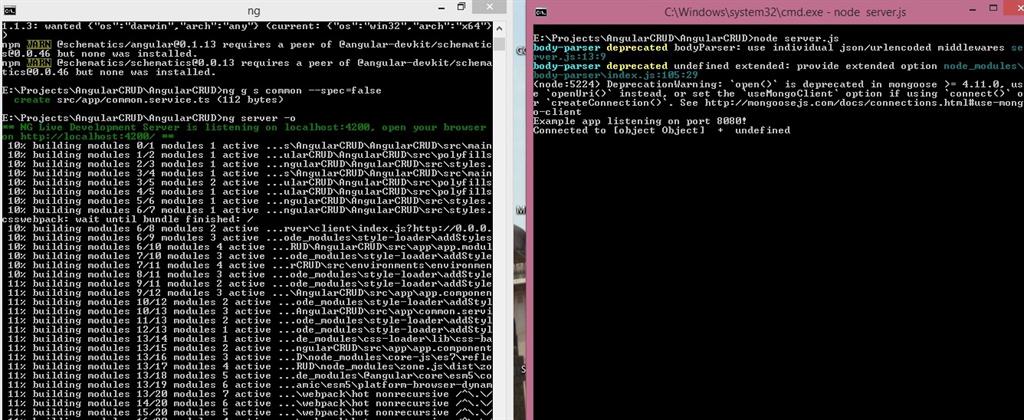
**Step 14**   
  
Write this code in app.component.ts and remove the existing code from this file.

2. **import** { Component, OnInit } from '@angular/core';
3. **import** {FormGroup,FormControl,Validators,FormsModule, } from '@angular/forms';
4. **import** {CommonService} from './common.service';
6. **import** {Http,Response, Headers, RequestOptions } from '@angular/http';
8. @Component({
9. selector: 'app-root',
10. templateUrl: './app.component.html',
11. styleUrls: ['./app.component.css']
12. })
13. **export** **class** AppComponent {

16. constructor(**private** newService :CommonService,) {   }
17. Repdata;
18. valbutton ="Save";

21. ngOnInit() {
22. **this**.newService.GetUser().subscribe(data =>  **this**.Repdata = data)
23. }
25. onSave = **function**(user,isValid: **boolean**) {
26. user.mode= **this**.valbutton;
27. **this**.newService.saveUser(user)
28. .subscribe(data =>  {  alert(data.data);
30. **this**.ngOnInit();
31. }
32. , error => **this**.errorMessage = error )
34. }
35. edit = **function**(kk) {
36. **this**.id = kk.\_id;
37. **this**.name= kk.name;
38. **this**.address= kk.address;
39. **this**.valbutton ="Update";
40. }
42. **delete** = **function**(id) {
43. **this**.newService.deleteUser(id)
44. .subscribe(data =>   { alert(data.data) ; **this**.ngOnInit();}, error => **this**.errorMessage = error )
45. }
47. }

We are almost done for performing select, insert, update, delete operation. Now, let us run two servers. The first one is Angular application with command  ng server-o and the second one is to open node.js server.



We seen the output on borwser port 4200.

# https://csharpcorner-mindcrackerinc.netdna-ssl.com/article/angular-5-crud-opration-using-nodeexpressmongo/Images/ag15.jpg CRUD Operation In Angular 6

we will be building an Angular 6 CRUD Operation application step by step from scratch with example. We will be generating our Angular 6 application using angular CLI and then modify it to have an employee management project where end user can perform CRUD operations such as create, list, update and delete with the sample REST API exposed using HttpClientModule. We will also be using RouterModule to have routing enabled.

**Angular 6 CRUD Operation**

In this article, we will be building an Angular 6 application and will perform a CRUD Operation step by step from scratch with an example. We will be generating our Angular 6 application using Angular CLI and then modify it to have an employee management project where the end-user can perform CRUD operations, i.e., Create, List, Update, and Delete with the sample REST API exposed using HttpClientModule. We will also be using RouterModule to have the routing enabled.

For this project, I have npm 5.6.0 and node v8.11.2 installed on my local system. You can download the latest version of Node from [here](https://nodejs.org/en/download/). To update NPM, you can run the following command in the terminal.

*npm i npm@latest -g*

If you have the @angular/cli version older than 6, then run the following command to install the latest version.

*npm uninstall -g @angular/cli  
npm cache clean  
npm install -g @angular/cli*

To install a specific version, you can use this command -

*npm install -g @angular/cli@6.1.6*

**Generating Angular 6 Project**

Once the npm and node are upgraded to the latest version, you can run the following command to generate an Angular 6 project in any location of your choice.

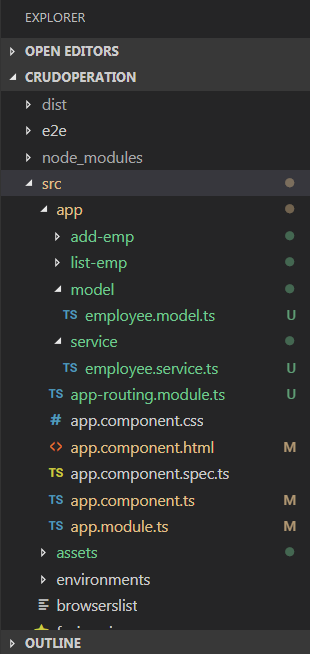
*ng new crudoperation*

Doing so, our Angular 6 application is generated.

**Angular 6 Project Structure**

Once the project is generated, you can run the following commands to see the Angular 6 app running at localhost:4200.

*cd crudoperation  
ng serve*

****

**Routing**

Following is our routing configuration. We have configured it to use ListEmpComponent as a default component. Also, do not forget to include it in the main module - app.module.ts.

**app-routing.module.ts**

1. **import** { NgModule } from '@angular/core';
2. **import** { CommonModule } from '@angular/common';
3. **import** { RouterModule, Routes } from '@angular/router';
4. **import** { ListEmpComponent } from './list-emp/list-emp.component';
5. **import** { AddEmpComponent } from './add-emp/add-emp.component';
7. **export** **const** routes: Routes = [
8. { path: '', component: ListEmpComponent, pathMatch: 'full' },
9. { path: 'list-emp', component: ListEmpComponent },
10. { path: 'add-emp', component: AddEmpComponent }
11. ];
13. @NgModule({
14. imports: [
15. CommonModule,
16. RouterModule.forRoot(routes)
17. ],
18. exports: [RouterModule],
19. declarations: []
20. })
21. **export** **class** AppRoutingModule { }

**app.module.ts**

1. **import** { BrowserModule } from '@angular/platform-browser';
2. **import** { NgModule } from '@angular/core';
3. **import** { HttpClientModule } from '@angular/common/http';
4. **import** { AppRoutingModule } from './app-routing.module';
5. **import** { ReactiveFormsModule } from "@angular/forms";
7. **import** { AppComponent } from './app.component';
8. **import** { ListEmpComponent } from './list-emp/list-emp.component';
9. **import** { AddEmpComponent } from './add-emp/add-emp.component';
10. **import** { EmployeeService } from './service/employee.service';
12. @NgModule({
13. declarations: [
14. AppComponent,
15. ListEmpComponent,
16. AddEmpComponent
17. ],
18. imports: [
19. BrowserModule,
20. HttpClientModule,
21. AppRoutingModule,
22. ReactiveFormsModule
23. ],
24. providers: [EmployeeService],
25. bootstrap: [AppComponent]
26. })
27. **export** **class** AppModule { }
29. Model
30. **export** **class** Employee {
31. id?: number;
32. employee\_name?: string;
33. employee\_salary?: number;
34. employee\_age?: number;
35. }

Following is the implementation of our EmployeeService. It has all the API details that are required for the CRUD operation. Here, I have used JSON Server for making API calls. The JSON Server is for front-end developers, which simulates a back-end REST Service to deliver the data in JSON format to the front-end application and make sure everything is working as expected.

1. **import** { Injectable } from '@angular/core';
2. **import** { HttpClient } from '@angular/common/http';
3. **import** { Employee } from '../model/employee.model';
5. @Injectable({
6. providedIn: 'root'
7. })
8. **export** **class** EmployeeService {
10. constructor(**private** http: HttpClient) { }
11. baseUrl: string = 'http://localhost:3004/posts/';
13. getEmployees() {
14. **return** **this**.http.get<Employee[]>(**this**.baseUrl);
15. }
16. deleteEmployees(id: number) {
17. **return** **this**.http.**delete**<Employee[]>(**this**.baseUrl + id);
18. }
19. createUser(employee: Employee) {
20. **return** **this**.http.post(**this**.baseUrl, employee);
21. }
22. getEmployeeById(id: number) {
23. **return** **this**.http.get<Employee>(**this**.baseUrl + '/' + id);
24. }
25. updateEmployee(employee: Employee) {
26. **return** **this**.http.put(**this**.baseUrl + '/' + employee.id, employee);
27. }
28. }

**Creating Components list-emp.component.html**

1. <div **class**="col-md-12">
2. <h2> User Details</h2>
3. <div **class**="table-responsive table-container">
4. <table **class**="table">
5. <thead>
6. <tr>
7. <th>Id</th>
8. <th>Employee Name</th>
9. <th>Salary</th>
10. <th>Age</th>
11. </tr>
12. </thead>
13. <tbody>
14. <tr \*ngFor="let emp of employees">
15. <td **class**="hidden">{{emp.id}}</td>
16. <td>{{emp.employee\_name}}</td>
17. <td>{{emp.employee\_salary}}</td>
18. <td>{{emp.employee\_age}}</td>
19. <td>
20. <button (click)="deleteEmp(emp)" **class**="btn btn-info"> Delete</button>
21. <button (click)="editEmp(emp)" style="margin-left: 20px;" **class**="btn btn-info"> Edit</button>
22. </td>
23. </tr>
24. </tbody>
25. </table>
26. </div>
27. </div>

**list-emp.component.ts**

1. **import** { Component, OnInit } from '@angular/core';
2. **import** { EmployeeService } from '../service/employee.service';
3. **import** { Employee } from '../model/employee.model';
4. **import** { Router } from "@angular/router";
6. @Component({
7. selector: 'app-list-emp',
8. templateUrl: './list-emp.component.html',
9. styleUrls: ['./list-emp.component.css']
10. })
11. **export** **class** ListEmpComponent **implements** OnInit {
13. employees: Employee[];
15. constructor(**private** empService: EmployeeService, **private** router: Router, ) { }
17. ngOnInit() {
18. **this**.empService.getEmployees()
19. .subscribe((data: Employee[]) => {
20. **this**.employees = data;
21. });
22. }
23. deleteEmp(employee: Employee): **void** {
24. **this**.empService.deleteEmployees(employee.id)
25. .subscribe(data => {
26. **this**.employees = **this**.employees.filter(u => u !== employee);
27. })
28. }
29. editEmp(employee: Employee): **void** {
30. localStorage.removeItem('editEmpId');
31. localStorage.setItem('editEmpId', employee.id.toString());
32. **this**.router.navigate(['add-emp']);
33. }
34. }

**add-emp.component.html**

1. <div **class**="col-md-6">
2. <h2 **class**="text-center">{{empformlabel}}</h2>
3. <form [formGroup]="addForm" novalidate **class**="form">
4. <div **class**="form-group">
5. <label **for**="empId">Employee Id:</label>
6. <input type="number" formControlName="id" placeholder="Id" name="empId" **class**="form-control" id="empId">
7. </div>
9. <div **class**="form-group">
10. <label **for**="empName">Employee Name:</label>
11. <input formControlName="employee\_name" placeholder="Employee Name" name="empName" **class**="form-control" id="empName">
12. <div **class**="alert  alert-danger" \*ngIf="addForm.get('employee\_name').hasError('required') && addForm.get('employee\_name').touched">
13. Employee Name is required
14. </div>
15. </div>
17. <div **class**="form-group">
18. <label **for**="empSalary">Employee Salary:</label>
19. <input formControlName="employee\_salary" placeholder="Employee Salary" name="employee\_salary" **class**="form-control" id="employee\_salary">
20. <div **class**="alert  alert-danger" \*ngIf="addForm.get('employee\_salary').hasError('maxlength') && addForm.get('employee\_salary').touched">
21. Employee Salary is required and should less than 9 characters.
22. </div>
23. </div>
25. <div **class**="form-group">
26. <label **for**="empAge">Employee Age:</label>
27. <input formControlName="employee\_age" placeholder="Employee Age" name="empAge" **class**="form-control" id="empAge">
28. <div **class**="alert  alert-danger" \*ngIf=" addForm.get('employee\_age').hasError('maxlength') && addForm.get('employee\_age').touched">
29. Age is required and should less than 3 characters.
30. </div>
31. </div>
32. <button **class**="btn btn-success" [disabled]='addForm.invalid' \*ngIf="btnvisibility" (click)="onSubmit()">Save</button>
33. <button **class**="btn btn-success" [disabled]='addForm.invalid' \*ngIf="!btnvisibility" (click)="onUpdate()">Update</button>
35. <p>Form value: {{ addForm.value | json }}</p>
36. </form>
37. </div>

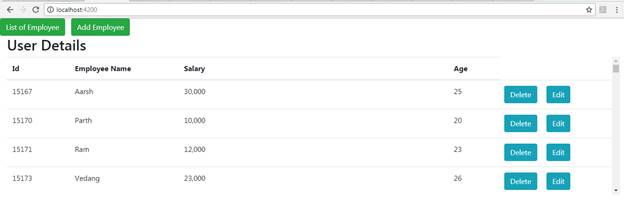
**add-emp.component.ts**

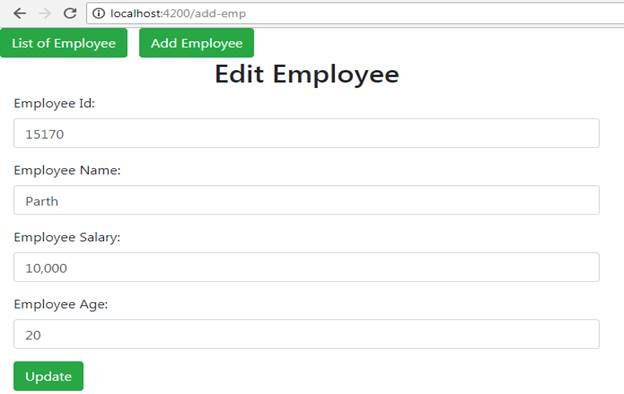
1. **import** { Component, OnInit } from '@angular/core';
2. **import** { FormBuilder, FormGroup, Validators } from "@angular/forms";
3. **import** { EmployeeService } from '../service/employee.service';
4. **import** { Router } from "@angular/router";
6. @Component({
7. selector: 'app-add-emp',
8. templateUrl: './add-emp.component.html',
9. styleUrls: ['./add-emp.component.css']
10. })
11. **export** **class** AddEmpComponent **implements** OnInit {
13. empformlabel: string = 'Add Employee';
14. empformbtn: string = 'Save';
15. constructor(**private** formBuilder: FormBuilder, **private** router: Router, **private** empService: EmployeeService) {
16. }
18. addForm: FormGroup;
19. btnvisibility: **boolean** = **true**;
20. ngOnInit() {
22. **this**.addForm = **this**.formBuilder.group({
23. id: [],
24. employee\_name: ['', Validators.required],
25. employee\_salary: ['', [Validators.required, Validators.maxLength(9)]],
26. employee\_age: ['', [Validators.required, Validators.maxLength(3)]]
27. });
29. let empid = localStorage.getItem('editEmpId');
30. **if** (+empid > 0) {
31. **this**.empService.getEmployeeById(+empid).subscribe(data => {
32. **this**.addForm.patchValue(data);
33. })
34. **this**.btnvisibility = **false**;
35. **this**.empformlabel = 'Edit Employee';
36. **this**.empformbtn = 'Update';
37. }
38. }
39. onSubmit() {
40. console.log('Create fire');
41. **this**.empService.createUser(**this**.addForm.value)
42. .subscribe(data => {
43. **this**.router.navigate(['list-emp']);
44. },
45. error => {
46. alert(error);
47. });
48. }
49. onUpdate() {
50. console.log('Update fire');
51. **this**.empService.updateEmployee(**this**.addForm.value).subscribe(data => {
52. **this**.router.navigate(['list-emp']);
53. },
54. error => {
55. alert(error);
56. });
57. }
58. }

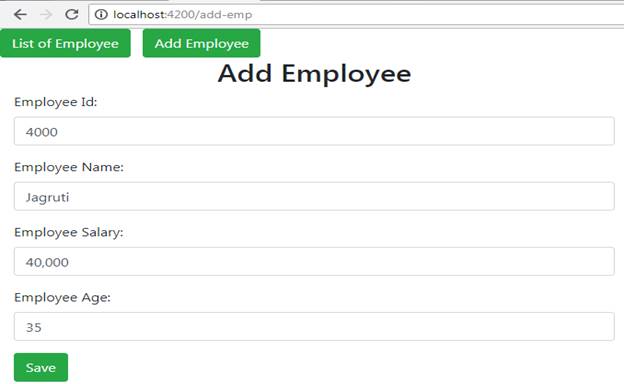
**Global Style style.css**

1. /\* You can add global styles to this file, and also import other style files \*/
2. @**import** "~bootstrap/dist/css/bootstrap.css";
3. @**import** "~font-awesome/css/font-awesome.css";
4. .ng-valid[required], .ng-valid.required  {
5. border-left: 5px solid #42A948; /\* green \*/
6. }
7. .ng-invalid:not(form)  {
8. border-left: 5px solid #a94442; /\* red \*/
9. }
10. .mtop10{
11. margin-top:10px;
12. }

**Testing Angular 6 Application**  
Now, run the command *ng serve*and hit localhost:4200.  
  
You can see the following screen with a list of users. On this page, you can perform actions to add, edit, and delete an employee.  







Here, we have used the same component to Add and Edit/Update the Employee.