Spring Boot with angular Rest A pi Example:-

Angular Front End step by step :-

Create angularpro Project folder and open in visual code you will see project structure like this :-

Open cmd and type command

ng new angualrpro as shown below

```
D:\nikita>ng new  angularpro
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE angularpro/angular.json (2720 bytes)
```

Now go to your project folder as shown below

```
D:\nikita>cd  angularpro

D:\nikita\angularpro>
```

Now add component adduser as shown below :-

```
D:\nikita>cd  angularpro

D:\nikita\angularpro>ng g c adduser
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

    ng analytics disable

Global setting: enabled
Local setting: enabled
Effective status: enabled
CREATE src/app/adduser/adduser.component.html (22 bytes)
CREATE src/app/adduser/adduser.component.spec.ts (566 bytes)
CREATE src/app/adduser/adduser.component.ts (206 bytes)
CREATE src/app/adduser/adduser.component.css (0 bytes)
UPDATE src/app/app.module.ts (479 bytes)
```

After it create one more component

userclass with command ng g c userclass as shown below

```
D:\nikita\angularpro>ng g c  userclass
CREATE src/app/userclass/userclass.component.html (24 bytes)
CREATE src/app/userclass/userclass.component.spec.ts (580 bytes)
CREATE src/app/userclass/userclass.component.ts (214 bytes)
CREATE src/app/userclass/userclass.component.css (0 bytes)
UPDATE src/app/app.module.ts (573 bytes)

D:\nikita\angularpro>
```

Now create one more component

toaster-demo with command ng g c toaster-demo as shown below.

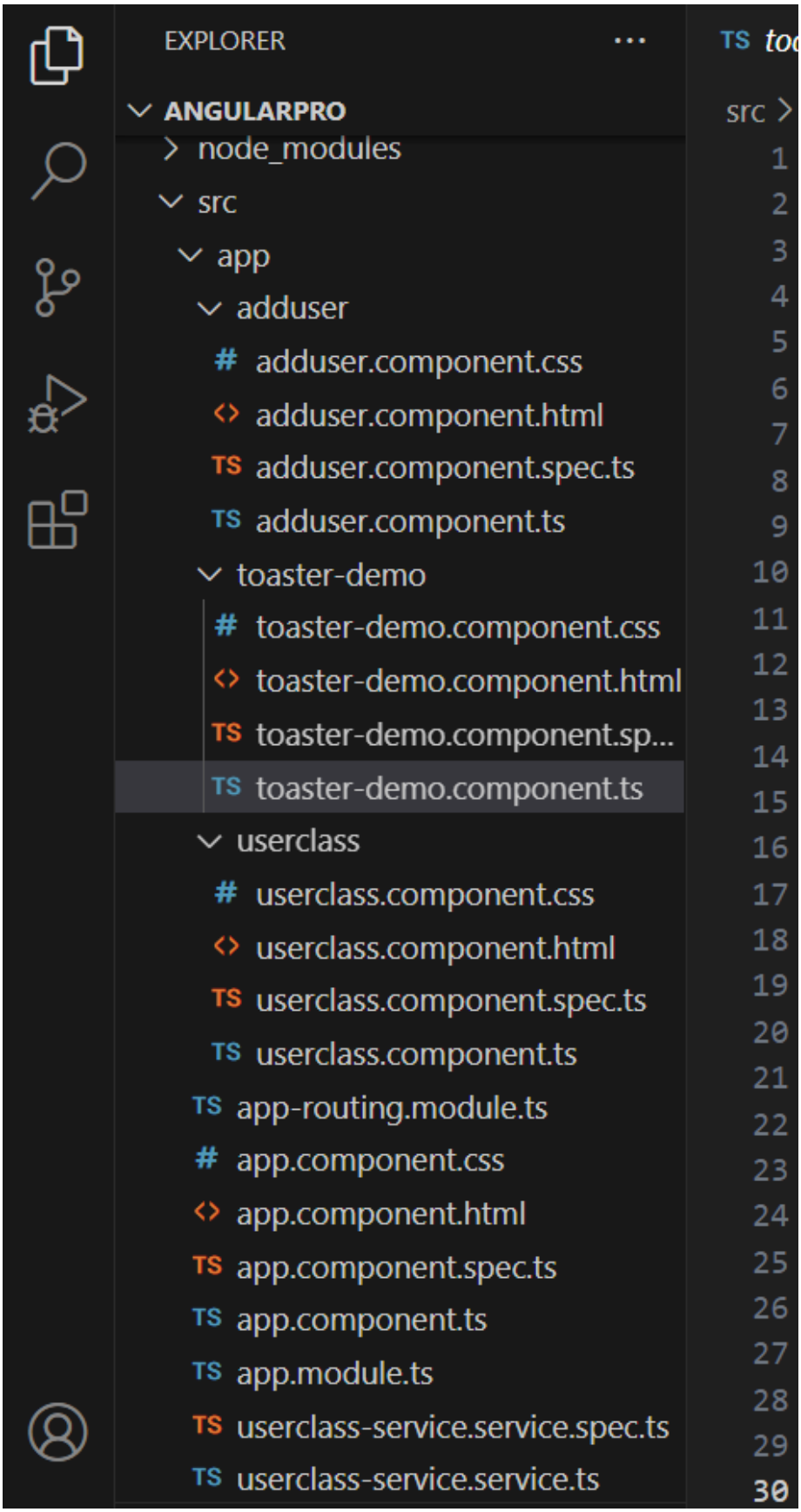
```
D:\nikita\angularpro>ng  g c toaster-demo
CREATE src/app/toaster-demo/toaster-demo.component.html (27 bytes)
CREATE src/app/toaster-demo/toaster-demo.component.spec.ts (595 bytes)
CREATE src/app/toaster-demo/toaster-demo.component.ts (225 bytes)
CREATE src/app/toaster-demo/toaster-demo.component.css (0 bytes)
UPDATE src/app/app.module.ts (677 bytes)
```

And create service userclass with command

ng generate service usercalss-service

```
D:\nikita\angularpro>ng generate service  userclass-service
CREATE src/app/userclass-service.service.spec.ts (408 bytes)
CREATE src/app/userclass-service.service.ts (145 bytes)

D:\nikita\angularpro>
```

Finally open your project in visual code you will see project structure like this.


EXPLORER
ANGULARPRO
node_modules
src
app
adduser
adduser.component.css
adduser.component.html
adduser.component.spec.ts
adduser.component.ts
toaster-demo
toaster-demo.component.css
toaster-demo.component.html
toaster-demo.component.sp...
toaster-demo.component.ts
userclass
userclass.component.css
userclass.component.html
userclass.component.spec.ts
userclass.component.ts
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
userclass-service.service.spec.ts
userclass-service.service.ts
src
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Create main app.component.html file:-

app.component.html

src > app > app.component.html > router-outlet

```
1
2   <!-- <app-toaster-demo></app-toaster-demo> -->
3   <!-- <app-adduser></app-adduser> -->
4
5   <!-- Navigates to /adduser page on button click -->
6   <button routerLink="/adduser">Registration1</button>
7   <button [routerLink]="['/adduser']">Registration2</button>
8
9   <a class="nav-link" routerLink="/adduser">Registration3</a>
10  <br>
11
12  <router-outlet></router-outlet>
```

app.module.ts :-

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { ToastrModule } from 'ngx-toastr';
import { animation } from '@angular/animations';
import { ToasterDemoComponent } from './toaster-demo/toaster-demo.component';
import { AdduserComponent } from './adduser/adduser.component';
import { HttpClientModule } from '@angular/common/http';
import { UserclassComponent } from './userclass/userclass.component';

@NgModule({
  declarations: [
```

```
    AppComponent,
    ToasterDemoComponent,
    AdduserComponent,
    UserclassComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
   BrowserAnimationsModule,
    ToastrModule.forRoot(),
    HttpClientModule

  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

TS app.component.ts

src > app > TS app.component.ts > ...

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'angularpro';
}

```

TS app-routing.module.ts

src > app > TS app-routing.module.ts > ...

```ts
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AdduserComponent } from './adduser/adduser.component';
import { UserclassComponent } from './userclass/userclass.component';

const routes: Routes = [
  {path:"users",component:UserclassComponent},
  {path:"adduser",component:AdduserComponent},
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }

```

Now create component adduser :-

adduser.component.html file:-

```
<div class="form-group">
    <label>Enter Username</label>
    <input placeholder="Enter Username" [(ngModel)]="u.username" class="form-
control"/>
</div>
<div class="form-group">
    <label>Enter Email</label>
    <input placeholder="Enter Email" [(ngModel)]="u.email" class="form-control"/>
</div>
<div class="form-group">
    <label>Enter Password</label>
    <input type="password" placeholder="Enter Password" [(ngModel)]="u.password"
class="form-control"/>
</div>
<div class="form-group">
    <button class="btn btn-primary" (click)="addUser()">Register</button>
</div>
```

adduser.component.ts file :-

```
import { Component, OnInit } from '@angular/core';
import { UserclassServiceService } from '../userclass-service.service';
import { UserClass } from '../userclass';
import { Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-adduser',
  templateUrl: './adduser.component.html',
  styleUrls: ['./adduser.component.css']
})
export class AdduserComponent implements OnInit {
  u=new UserClass();

  constructor( private userClassService:UserclassServiceService,private
toastr:ToastrService,
    private router:Router )
  {}

  addUser()
```

```
  {
    this.u.role="user";
    this.userClassService.addUser(this.u).subscribe((result)=>
    {
      this.toastr.success("User Added Successfully!!!");
      this.router.navigate(["/users"]);
    });
  }


  ngOnInit(): void {
  }

}
```

Now create userclass component:-

userclass.component.html file :-

```
<h2>UserClass Details!</h2>
<table class="table table-bordered">
    <Tr>
        <th>User Id</th>
        <th>UserName</th>
        <th>Email</th>
        <th>Password</th>
        <th>Role</th>
    </Tr>
    <tr *ngFor="let user of users">
        <td>{{user.userid}}</td>
        <td>{{user.username}}</td>
        <td>{{user.email}}</td>
        <td>{{user.password}}</td>
        <td *ngIf="user.role">{{user.role}}</td>
        <td *ngIf="!user.role">No Role</td>
        <td>
            <button class="btn btn-warning" (click)="edit(user)">Edit</button>
            <button class="btn btn-danger" (click)="delete(user)">Delete</button>
        </td>
    </tr>
</table>
```

```
<div *ngIf="selectedUser">
    <div class="form-group">
        <label hidden>Enter Userid</label>
        <input placeholder="Enter UserId" hidden
[(ngModel)]="selectedUser.userid" class="form-control"/>
    </div>
    <div class="form-group">
        <label>Enter Username</label>
        <input placeholder="Enter Username" [(ngModel)]="selectedUser.username"
class="form-control"/>
    </div>
    <div class="form-group">
        <label>Enter Email</label>
        <input placeholder="Enter Email" [(ngModel)]="selectedUser.email"
class="form-control"/>
    </div>
    <div class="form-group">
        <label>Enter Password</label>
        <input placeholder="Enter Password" [(ngModel)]="selectedUser.password"
class="form-control"/>
    </div>
    <div class="form-group">
        <button (click)="updateUser()" class="btn btn-warning">Update
User</button>
        <button (click)="cancel()" class="btn btn-primary">Cancel</button>
    </div>
</div>
```

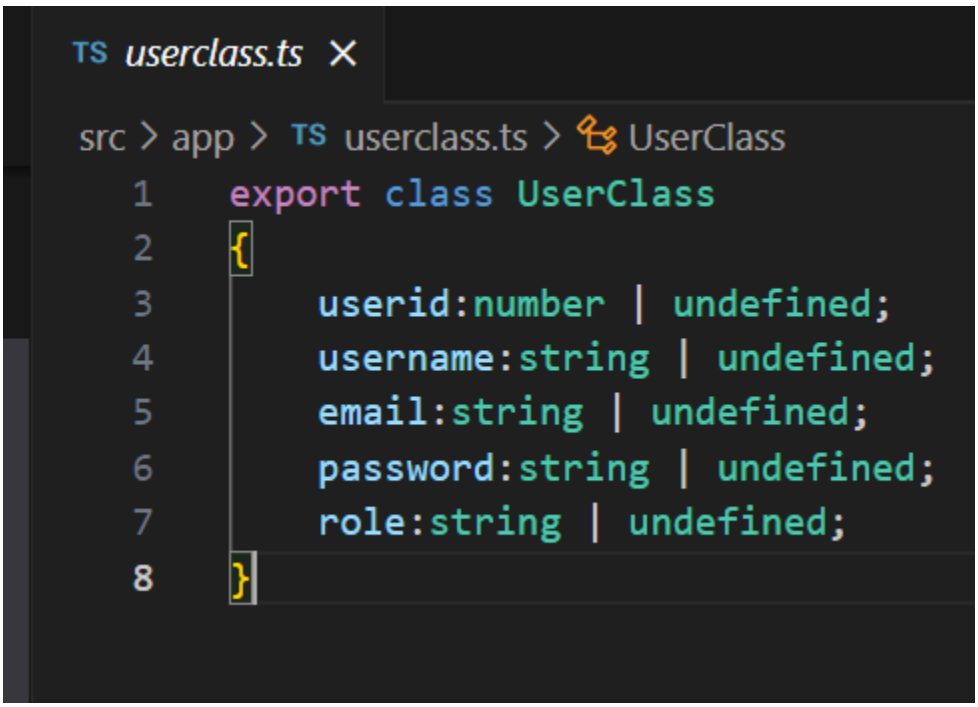
userclass.component.ts file :-

```
import { Component, OnInit } from '@angular/core';
import { UserClass } from '../userclass';
import { UserclassServiceService } from '../userclass-service.service';
import { ToastrService } from 'ngx-toastr';
import { Router } from '@angular/router';

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

```
  users:UserClass[]=[];
  selectedUser:UserClass | undefined;

constructor(private userClassService:UserclassServiceService,
  private toastr:ToastrService,
  private router:Router){}

  getUserDetails()
  {
    this.userClassService.getUserDetails().subscribe(result=>
      {
        this.users=result
      },
      (err: { message: string | undefined; })=>
      {
          this.toastr.error(err.message);
      }
      );
  }

  edit(u:UserClass)
  {
    this.selectedUser=u;
  }

  cancel()
  {
    this.selectedUser=undefined;
  }

  updateUser()
  {
    this.userClassService.updateUser(this.selectedUser).subscribe((result)=>
    {
      this.toastr.success("User Updated successfully!!!");
    });
  }

  delete(u:UserClass)
  {
    this.userClassService.deleteUser(u).subscribe((result)=>
    {
      this.toastr.success("User Deleted successfully!!!");
      this.getUserDetails();
    },
```

```
    (err)=>
    {
      this.toastr.error(err.message);
    });
  }

  ngOnInit(): void {
   this.getUserDetails();
  }
}
```

Now create userclass.ts file :-



now create userclassserviceservice

userclass-service.service.ts file code:-

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { UserClass } from './userclass';

@Injectable({
  providedIn: 'root'
})
export class UserclassServiceService {

  constructor(private http:HttpClient) { }

  getUserDetails()
  {
    return this.http.get<UserClass[]>("http://localhost:8080/user/");
  }
  addUser(u:UserClass)
  {
    return this.http.post("http://localhost:8080/user/add",u);
  }
  updateUser(u:UserClass | undefined)
  {
    return this.http.put("http://localhost:8080/user/update",u);
  }
  deleteUser(u:UserClass)
  {
    return this.http.delete("http://localhost:8080/user/delete/"+u.userid)
  }
}
```

Create toaster-demo component:-

toaster-demo.component.html

src > app > toaster-demo > toaster-demo.component.html > button

```html
<h1>Angular Toastr Notifications Example</h1>

<button (click)="showToasterSuccess()">
    Success Toaster
</button>

<button (click)="showToasterError()">
    Error Toaster
</button>

<button (click)="showToasterInfo()">
    Info Toaster
</button>

<button (click)="showToasterWarning()">
    Warning Toaster
</button>
```

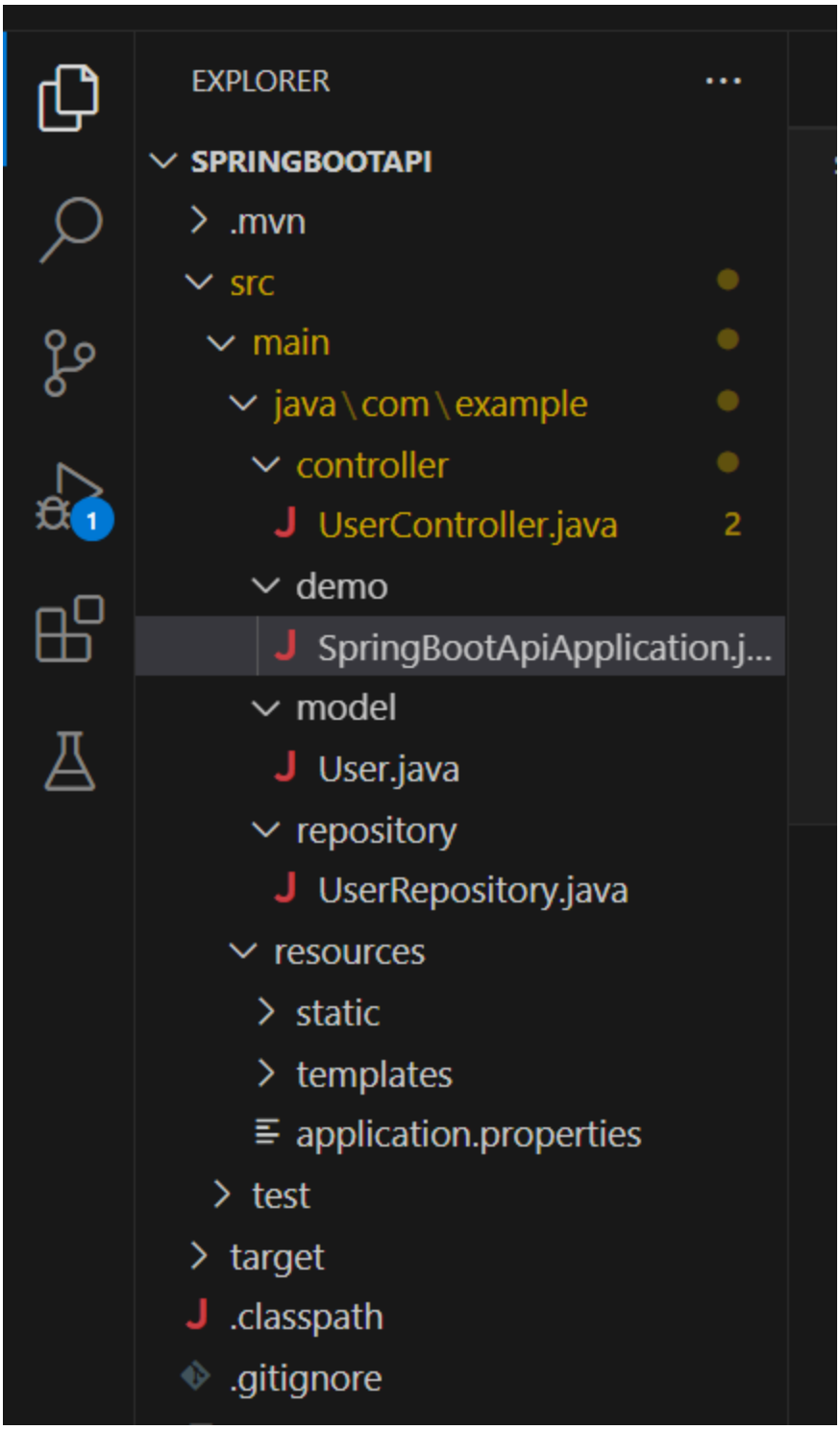
Write code for toaster-demo.component.ts file code :-

```typescript
import { Component } from '@angular/core';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-toaster-demo',
  templateUrl: './toaster-demo.component.html',
  styleUrls: ['./toaster-demo.component.css']
})
export class ToasterDemoComponent {
  title = 'toaster-not';

  constructor(private toastr: ToastrService) { }
```

```
  showToasterSuccess(){
      this.toastr.success("Data shown successfully !!",this.title)
  }

  showToasterError(){
      this.toastr.error("Something is wrong", this.title)
  }

  showToasterInfo(){
      this.toastr.info("This is info", this.title)
  }

  showToasterWarning(){
      this.toastr.warning("This is warning", this.title)
  }
}
```

Spring boot rest Api Backend Development :-

Project structure will be like this –



Write code for model User.java file code:-

```
package com.example.model;

import javax.persistence.Column;
```

```java
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="users")
public class User
{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userid;

    @Column
    private String username;
    private String email;
    private String password;
    private String role;

    public int getUserid() {
        return userid;
    }
    public void setUserid(int userid) {
        this.userid = userid;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public String getRole() {
```

```
        return role;
    }
    public void setRole(String role) {
        this.role = role;
    }



}
```

Now write code for `UserRepository.java` file code:-

```
package com.example.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.example.model.User;

@Repository
public interface UserRepository extends JpaRepository<User, Integer>
{
    public User findByUsername(String name);
    public User findByEmail(String name);
    public User findByUserid(int id);
}
```

Now write code for controller class file `UserController.java` file:-

```
package com.example.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
```

```java
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.model.User;
import com.example.repository.UserRepository;

@RestController
@RequestMapping("/user")
@CrossOrigin(value = "http://localhost:4200/")
public class UserController
{
    @Autowired
    UserRepository userRepo;

    @GetMapping("/")
    public List<User> getUsers()
    {
        return userRepo.findAll();
    }

    @GetMapping("/username/{name}")
    public User getUsers(@PathVariable("name") String name)
    {
        return userRepo.findByUsername(name);
    }

    @GetMapping("/email/{name}")
    public User getEmail(@PathVariable("name") String name)
    {
        return userRepo.findByEmail(name);
    }

    @GetMapping("/id/{id}")
    public User getEmail(@PathVariable("id") int id)
    {
        return userRepo.findByUserid(id);
    }

    @PostMapping("/add")
    public void addUser(@RequestBody User u)
    {
        userRepo.save(u);
    }
```

```
    @PutMapping("/update")
    public void updateUser(@RequestBody User u)
    {
        userRepo.save(u);
    }

    @DeleteMapping("/delete/{id}")
    public void updateUser(@PathVariable("id") int id)
    {
        User u=new User();
        u.setUserid(id);
        userRepo.delete(u);
    }
}
```

Write following code in application.properties file which you will find under resources folder

```
spring.datasource.url=jdbc:mysql://localhost:3306/mydb
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

server.error.whitelabel.enabled=false
spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect
```

and your application code SpringBootApiApplication.java file

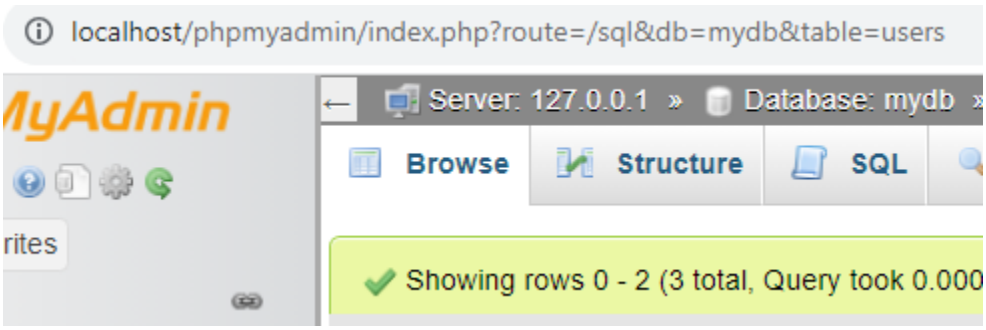
:-

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@SpringBootApplication
@ComponentScan(basePackages = "com.example")
@EntityScan("com.example.model")
```

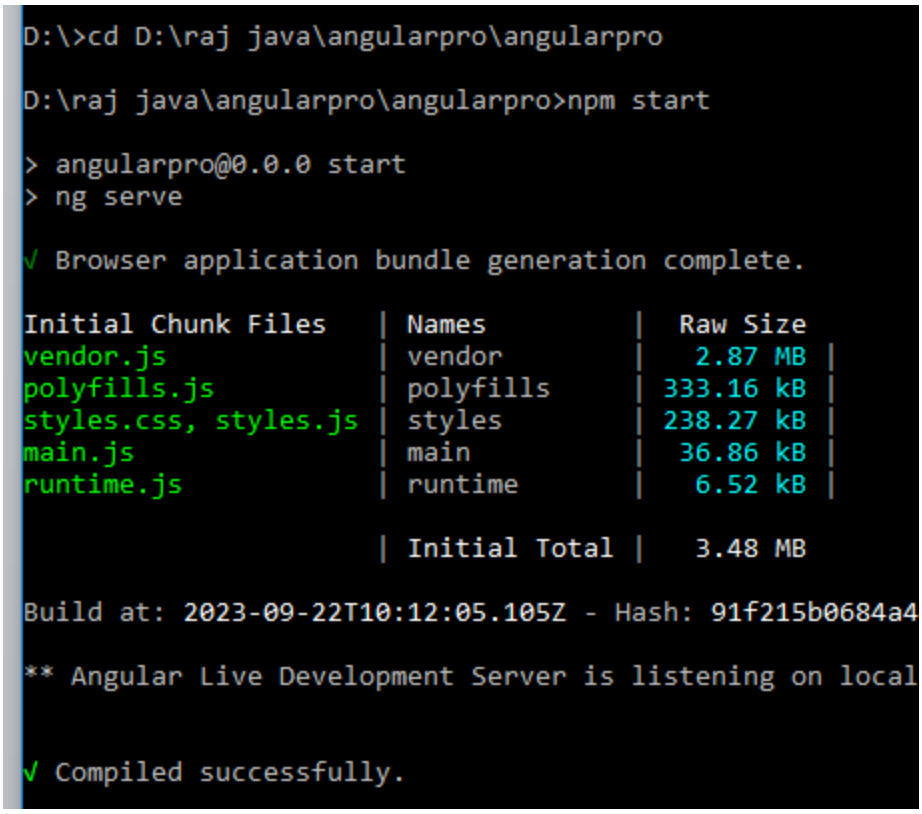
```java
@EnableJpaRepositories("com.example.repository")
public class SpringBootApiApplication
{
    public static void main(String[] args)
    {
        SpringApplication.run(SpringBootApiApplication.class, args);
    }

}
```

Now install xampp and run it and

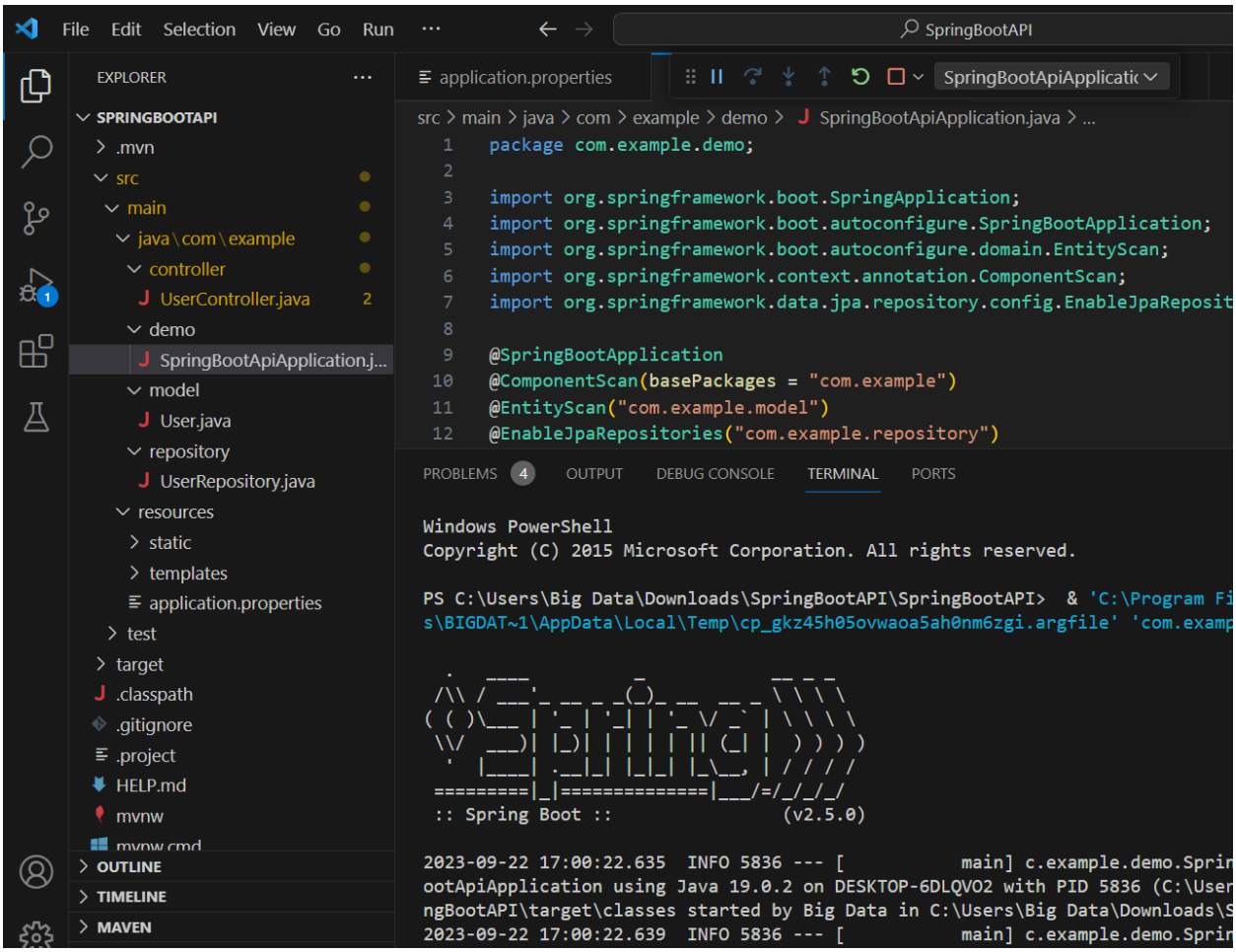
inside your "localhost/phpmyadmin" create database mydb as shown below



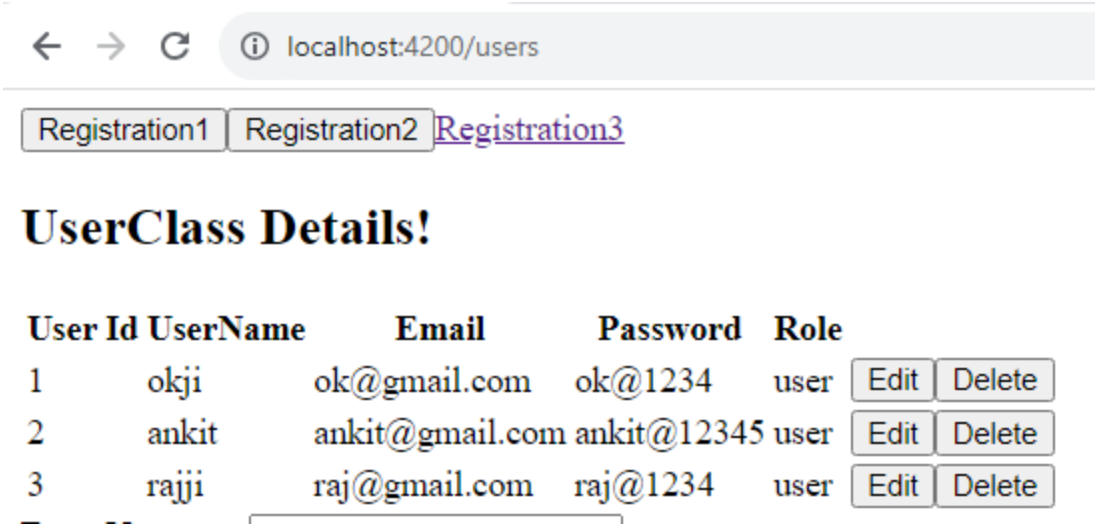
Finally run your front end angular code



and back end spring boot code :-



And open your browser you will see



After click on Registration 1 you will see