**ICIN FRONTEND SOURCE CODE**

**DEVELOPER NAME – SURAJ DUBEY**

Admin-Portal>adminportal-angular>src>app>authorize-registration

```
.alignment{
    text-align: center;
}
@media
only screen and (max-width: 1000px) {
  table, thead, tbody, th, td, tr,nav {
    display: block;
  }

  thead tr {
    position: absolute;
    top: -9999px;
    left: -9999px;
  }
  tr { border: 1px solid #ccc; }
  td {
    border: none;
    border-bottom: 1px solid #eee;
    position: relative;
    padding-left: 100px;
    margin-left: 150px;
  }
  td:before {
    position: absolute;
    top: 12px;
    left: 6px;
    width: 200px;
    padding-right: 40px;
    white-space: nowrap;
    margin-left: -150px;
  }


  td:nth-of-type(1):before { content: "First Name"; }
  td:nth-of-type(2):before { content: "Last Name"; }
  td:nth-of-type(3):before { content: "Phone Number"; }
  td:nth-of-type(4):before { content: "Address"; }
  td:nth-of-type(5):before { content: "Date of Birth"; }
  td:nth-of-type(6):before { content: "Type of Verification Document"; }
  td:nth-of-type(7):before { content: "Verification ID"; }
  td:nth-of-type(8):before { content: "Create Account"; }
  td:nth-of-type(9):before { content: "Cancel Account Creation"; }
```

```
}
```

```
<table class="table table-bordered">
  <thead>
    <tr>
      <th class="alignment">First Name</th>
      <th class="alignment">Last Name</th>
      <th class="alignment">Phone Number</th>
      <th class="alignment">Address</th>
      <th class="alignment">Date of Birth</th>
      <th class="alignment">Type of Verification Document</th>
      <th class="alignment">Verification ID No.</th>
      <th class="alignment">Email</th>
      <th class="alignment">Create Account</th>
      <th class="alignment">Cancel Account Creation</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let authorizeuser of authorizeusers">
      <td class="alignment">{{authorizeuser.fname}}</td>
      <td class="alignment">{{authorizeuser.lname}}</td>
      <td class="alignment">{{authorizeuser.phone}}</td>
      <td class="alignment">{{authorizeuser.address}}</td>
      <td class="alignment">{{authorizeuser.dob}}</td>
      <td class="alignment">{{authorizeuser.identityType}}</td>
      <td class="alignment">{{authorizeuser.identity}}</td>
      <td class="alignment">{{authorizeuser.email}}</td>
      <td class="alignment"><button type="button" class="btn btn-success"
(click)="authorizeAccount(authorizeuser.username)">Create</button> </td>
      <td class="alignment"><button type="button" class="btn btn-
danger"(click)="rejectRequest(authorizeuser.username)">Cancel</button></td>
    </tr>


  </tbody>
</table>
```

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { AuthorizeRegistrationComponent } from './authorize-
registration.component';

describe('AuthorizeRegistrationComponent', () => {
```

```
  let component: AuthorizeRegistrationComponent;
  let fixture: ComponentFixture<AuthorizeRegistrationComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ AuthorizeRegistrationComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(AuthorizeRegistrationComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

```
import { Component, OnInit } from '@angular/core';
import { AuthorizationService } from '../authorization.service'
import { AuthorizeUser } from '../model/authorizeUser';

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

  authorizeusers:AuthorizeUser[];
  constructor(public authorizeService: AuthorizationService) {

  }

  ngOnInit(){
    this.authorizeService.getRequestData().subscribe(res=>{
      console.log(res);
      this.authorizeusers = res});
  }

  authorizeAccount(username){
```

```
    this.authorizeService.authorizeAccount(username).subscribe(res=>this.ngOnI
nit());
  }

  rejectRequest(username){
    this.authorizeService.rejectRequest(username).subscribe(res=>this.ngOnInit
());
  }


}
```

Admin-Portal>adminportal-angular>src>app>checkbook-request

```
@media

only screen and (max-width: 1000px) {

  table, thead, tbody, th, td, tr,nav {

    display: block;

  }


  thead tr {

    position: absolute;

    top: -9999px;

    left: -9999px;

  }

  tr { border: 1px solid #ccc; }

  td {

    border: none;

    border-bottom: 1px solid #eee;

    position: relative;

    padding-left: 100px;

    margin-left: 150px;

  }

  td:before {

    position: absolute;
```

```
    top: 12px;
    left: 6px;
    width: 200px;
    padding-right: 40px;
    white-space: nowrap;
    margin-left: -150px;
  }
  td:nth-of-type(1):before { content: "Request Number"; }
  td:nth-of-type(2):before { content: "Type of Account"; }
  td:nth-of-type(3):before { content: "Account Number"; }
  td:nth-of-type(4):before { content: "Applied Date"; }
  td:nth-of-type(5):before { content: "Number of Pages"; }
  td:nth-of-type(6):before { content: " Confirm Request ?"; }



}
<table class="table table-bordered-md" style="overflow-x:auto;">
    <thead>
      <tr style="text-align: center">
        <th>Request Number</th>
        <th>Type of Account</th>
        <th>Account Number</th>
        <th>Applied Date</th>
        <th>Number of Pages</th>
        <th> Confirm Request ?</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let checkbookrequest of checkbookrequests | filter:term"
style="text-align: center">
        <td>{{checkbookrequest.id}}</td>
        <td>{{checkbookrequest.accType}}</td>
```

```
        <td>{{checkbookrequest.account}}</td>
        <td>{{checkbookrequest.date}}</td>
        <td>{{checkbookrequest.no_of_pages}}</td>
        <td><button (click)="confirmRequest(checkbookrequest.account)"
class="btn btn-primary">Confirm Request</button></td>

      </tr>

    </tbody>
  </table>
```

Ts

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { CheckbookRequestsComponent } from './checkbook-requests.component';

describe('CheckbookRequestsComponent', () => {
  let component: CheckbookRequestsComponent;
  let fixture: ComponentFixture<CheckbookRequestsComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ CheckbookRequestsComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
```

```
    fixture = TestBed.createComponent(CheckbookRequestsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Ts

```
import { Component, OnInit } from '@angular/core';
import { CheckbookService } from '../checkbook.service'
import { CheckbookRequest } from '../model/checkbookRequest';
import { NavigationEnd } from '@angular/router';

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

  checkbookrequests: CheckbookRequest[];
  term: string;


  constructor(public checkbookService: CheckbookService) {

  }
```

```
  ngOnInit() {
    this.checkbookService.getRequestsData().subscribe(res => {
      this.checkbookrequests = res
    });
  }

  getData() {

  }

  confirmRequest(account) {
    this.checkbookService.confirmCheckbookService(account).subscribe(res=>this
.ngOnInit());
  }

}
```

App>guards

Ts

```
import { TestBed } from '@angular/core/testing';

import { AuthGuard } from './auth.guard';

describe('AuthGuard', () => {
  let guard: AuthGuard;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    guard = TestBed.inject(AuthGuard);
  });
```

```
  it('should be created', () => {
    expect(guard).toBeTruthy();
  });
});
```

Ts

```
import { Injectable } from '@angular/core';
import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, UrlTree, Router } from '@angular/router';
import { Observable } from 'rxjs';
import { AuthenticationService } from '../service/authentication/authentication.service';


@Injectable({
  providedIn: 'root'
})
export class AuthGuard implements CanActivate {
  constructor(private authenticationService:AuthenticationService,private router:Router){


  }
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if(this.authenticationService.isAuthenticated){
      return true;
    }
      this.router.navigate(['']);
      return false;
  }


}
```

App>login

Css

```
.container{
   margin: 10rem auto;
   width:20em;
}

input{
    margin-bottom: 2rem ;
    margin-top: 2rem;
}

html{
    background-image: linear-gradient(#009933,white);
}
```

Html

```
<div class="container">
<h1>Admin Portal</h1>
<form class="form-login"
(ngSubmit)="onSubmit(loginForm)"  #loginForm="ngForm">

    <label for="inputUserName" class="sr-only">UserName</label>
    <input type="text" id="inputUserName" name="inputUserName" class="form-
control"  [ngClass]="{ 'is-invalid': submitted && inputUserName.errors }"
ngModel placeholder="UserName" required autofocus #inputUserName= ngModel>
    <div *ngIf="submitted && inputUserName.errors" class="invalid-feedback">
      <span *ngIf="inputUserName.errors.required">Please enter a
UserName</span>
    </div>
```

```
    <label for="inputPassword" class="sr-only">Password</label>
    <input type="password" id="inputPassword" name="password" class="form-control" [ngClass]="{ 'is-invalid': submitted && password.errors }" ngModel placeholder="Password" required #password=ngModel>
    <div *ngIf="submitted && password.errors" class="invalid-feedback">
      <span *ngIf="password.errors.required">Please enter a Password</span>
    </div>
    <br>
    <button class="btn btn-lg btn-primary btn-block" type="submit">Login</button>
  </form>
</div>
```

Ts

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { LoginComponent } from './login.component';

describe('LoginComponent', () => {
  let component: LoginComponent;
  let fixture: ComponentFixture<LoginComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ LoginComponent ]
    })
    .compileComponents();
  }));
```

```
  beforeEach(() => {
    fixture = TestBed.createComponent(LoginComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import {AuthenticationService} from
'../service/authentication/authentication.service';
import { LoginData } from '../model/loginData';
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {

  constructor(private authenticationService : AuthenticationService) { }
  submitted=false;
  ngOnInit(): void {
  }



  onSubmit(loginForm :NgForm)
```

```
{
  this.submitted=true;
 const Logindata = new
LoginData(loginForm.value.inputUserName,loginForm.value.password);
 this.authenticationService.authenticate(Logindata);
}


}
```

App>model

Ts

```
export class AuthorizeUser{
    fname:string;
    lname: string;
    phone: number;
    address: string;
    pan: string;
    email: string;
    username: string;
}
```

Ts

```
export class CheckbookRequest{
    id:number;
    accType: string;
    username: string;
    account: number;
    date: string;
    no_of_pages: number;
}
```

Ts

```
export class LoginData{

    username:string;

    password: string;


constructor (username:string, password:string){
this.username= username;

this.password= password;

}


}
```

Ts

```
export class UserData{

    fname:string;

    lname: string;

    username: string;

    email: string;

    account: number;

    saccount: number;

}
```

App.service/authenticaton

Ts

```
import { TestBed } from '@angular/core/testing';


import { AuthenticationService } from './authentication.service';


describe('AuthenticationService', () => {

  let service: AuthenticationService;


  beforeEach(() => {

    TestBed.configureTestingModule({});
```

```
  service = TestBed.inject(AuthenticationService);
});

it('should be created', () => {
```

```
    expect(service).toBeTruthy();
  });
});
```

Ts

```
import { Injectable } from '@angular/core';
import { LoginData } from 'src/app/model/loginData';
import { Router } from '@angular/router';


@Injectable({
  providedIn: 'root'
})


export class AuthenticationService {
private readonly adminUser = new LoginData('madhuri','mad12345');
isAuthenticated=false;
  constructor(private router: Router) { }


  authenticate(login: LoginData):boolean{
    if(this.checkCredentials(login)){
      this.isAuthenticated=true;
      this.router.navigate(['../user-account']);
      return true;
    }
    alert("Incorrect Login or Password");
    this.router.navigate(['../']);
```

```
    this.isAuthenticated = false;
    return false;

  }

checkCredentials(login:LoginData):boolean {
return this.checkEmail(login.username) && this.checkPassword(login.password);
}

checkEmail(email:string):boolean{
  return email=== this.adminUser.username;
}

checkPassword(password:string):boolean{
  return password=== this.adminUser.password;
}

logout(){
  this.isAuthenticated=false;
  this.router.navigate(['']);
}

}
```

App>user-account

Css

```
@media
only screen and (max-width: 1000px) {
  table, thead, tbody, th, td, tr,nav {
    display: block;
  }
```

```
thead tr {
  position: absolute;
  top: -9999px;
  left: -9999px;
}
th{
  text-align: center;
}
tr { border: 1px solid #ccc; }
td {
  border: none;
  border-bottom: 1px solid #eee;
  position: relative;
  padding-left: 100px;
  margin-left: 150px;
}
td:before {
  position: absolute;
  top: 12px;
  left: 6px;
  width: 200px;
  padding-right: 40px;
  white-space: nowrap;
  margin-left: -150px;
}
td:nth-of-type(1):before { content: "First Name"; }
td:nth-of-type(2):before { content: "Last Name"; }
td:nth-of-type(3):before { content: "UserName"; }
td:nth-of-type(4):before { content: "Current Account Number"; }
td:nth-of-type(5):before { content: "Savings Account Number"; }
```

```
  td:nth-of-type(6):before { content: "Enable Account"; }
  td:nth-of-type(7):before { content: "Disable Account"; }
  td:nth-of-type(8):before { content: "Features Granted to the User"; }
  td:nth-of-type(9):before { content: "Change Features"; }

}

  .select-selected {
    background-color: DodgerBlue;
  }

  /* Style the arrow inside the select element: */
  .select-selected:after {
    position: absolute;
    content: "";
    top: 14px;
    right: 10px;
    width: 0;
    height: 0;
    border: 6px solid transparent;
    border-color: #fff transparent transparent transparent;
  }

  /* Point the arrow upwards when the select box is open (active): */
  .select-selected.select-arrow-active:after {
    border-color: transparent transparent #fff transparent;
    top: 7px;
  }

  /* style the items (options), including the selected item: */
  .select-items div,.select-selected {
```

```
    color: #ffffff;
    padding: 8px 16px;
    border: 1px solid transparent;
    border-color: transparent transparent rgba(0, 0, 0, 0.1) transparent;
    cursor: pointer;
  }

  /* Style items (options): */
  .select-items {
    position: absolute;
    background-color: DodgerBlue;
    top: 100%;
    left: 0;
    right: 0;
    z-index: 99;
  }

  /* Hide the items when the select box is closed: */
  .select-hide {
    display: none;
  }

  .select-items div:hover, .same-as-selected {
    background-color: rgba(0, 0, 0, 0.1);
  }
```

Html

```
<table class="table table-bordered-md" style="overflow-x:auto;">
  <thead>
    <tr style="text-align: center">
      <th>First Name</th>
```

```
      <th>Last Name</th>
      <th style=" padding-bottom: 1.5%;">UserName</th>
      <!--<th>Email</th>-->
      <th>Primary Account Number</th>
      <th>Savings Account Number</th>
      <th>Enable Account</th>
      <th>Disable Account</th>
      <th style=" padding-bottom: 1.5%;">Features Granted to the User</th>
       <th style="padding-bottom: 1.5%;">Change Features</th>


    </tr>
  </thead>
  <tbody>
          <tr *ngFor="let user of users" style="text-align: center">
            <td>{{user.fname}}</td>
            <td>{{user.lname}}</td>
            <td>{{user.username}}</td>
            <td>{{user.primaryAccno}}</td>
            <td>{{user.savingsAccno}}</td>
            <td><button (click)="enableLoginService(user.username)" class="btn
btn-success">Enable</button></td>
            <td><button (click)="disableLoginService(user.username)"
class="btn btn-success">Disable</button></td>
            <td>{{user.featureStatus}}</td>
        <td style="text-align: center">


          <select (change) = "filterSelected($event.target.value)" >
            <option *ngFor="let r of roles" [value]='r.value'>
              {{r.name}}
            </option>
          </select>
```

```
          <button (click) = "setOption(user.username)" class="btn btn-
success">Set</button>



      </td>
          </tr>



        </tbody>
      </table>
```

Ts

```
import { async, ComponentFixture, TestBed } from '@angular/core/testing';

import { UserAccountComponent } from './user-account.component';

describe('UserAccountComponent', () => {
  let component: UserAccountComponent;
  let fixture: ComponentFixture<UserAccountComponent>;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [ UserAccountComponent ]
    })
    .compileComponents();
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(UserAccountComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });
```

```
  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Ts

```
import { Component, OnInit } from '@angular/core';
import { EnableService } from '../enable.service'
import { DisableService } from '../disable.service';
import { FeaturesService } from '../features.service';
import { UsersService } from '../users.service';
import { UserData } from '../model/userData';

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

  roles = [
    {name: " ", value:0},
    { name: "Deposit Access", value: 1 },
    { name: "Deposit + Withdraw Access", value: 2 },
    { name: "Deposit + Withdraw + Transfer Access", value: 3 }
  ]

  users: any[]
  //UserData[];
```

```
  constructor(public enableService: EnableService, public disableService:
DisableService, public featuresService: FeaturesService, private service:
UsersService) {


  }
  //selectedOption: string;
  //printedOption: number;
  selectedValue: number;


  ngOnInit() {
    this.service.getAllUsers().subscribe(res => {
      console.log(res)
     //console.log(res[0].featureStatus)
     res.forEach(element => {


       console.log(element.featureStatus)
      // console.log(this.roles[1].name)
       if(element.featureStatus == 1){
         element.featureStatus = this.roles[1].name
       }
       if (element.featureStatus == 2) {
        element.featureStatus = this.roles[2].name
       }
       if (element.featureStatus == 3) {
        element.featureStatus = this.roles[3].name
       }
     });
     this.users = res


    });
  }
```

```
  // print() {
  //   // this.printedOption = this.selectedOption;
  //   // console.log(this.selectedOption);
  // }

  filterSelected(selectedValue){
    this.selectedValue = selectedValue
    console.log('selected value= '+selectedValue);
  }

  enableLoginService(username) {
    console.log(username)
    this.enableService.enableLoginService(username).subscribe(res =>
this.ngOnInit());
    //this.enableService.enableLoginService();
  }

  disableLoginService(username) {
    this.disableService.disableLoginService(username).subscribe(res =>
this.ngOnInit());
    //this.disableService.disableLoginService();
  }

  setOption(username) {

    this.featuresService.setFeatures(username,
this.selectedValue).subscribe(res => this.ngOnInit());
    //this.featuresService.setFeatures();
  }


}
```

App>app-routing.module.ts

```
import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { LoginComponent } from './login/login.component';

import { UserAccountComponent } from './user-account/user-account.component';

import { CheckbookRequestsComponent } from './checkbook-requests/checkbook-
requests.component';

import {AuthGuard} from './guards/auth.guard'

import { AuthorizeRegistrationComponent } from './authorize-
registration/authorize-registration.component';

const routes: Routes = [

  {path: '', component:LoginComponent},

  {path:'user-account',
component:UserAccountComponent,canActivate:[AuthGuard]},

  {path:'checkbook-
requests',component:CheckbookRequestsComponent,canActivate:[AuthGuard]},

  {path:'authorize',component:AuthorizeRegistrationComponent,canActivate:[Auth
Guard]}

];


@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})


export class AppRoutingModule { }
```

app>app.component.spec.ts

```
import { TestBed, async } from '@angular/core/testing';

import { RouterTestingModule } from '@angular/router/testing';

import { AppComponent } from './app.component';


describe('AppComponent', () => {
```

```
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      imports: [
        RouterTestingModule
      ],
      declarations: [
        AppComponent
      ],
    }).compileComponents();
  }));

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'template-app'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('template-app');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement;
    expect(compiled.querySelector('.content
span').textContent).toContain('template-app app is running!');
  });
});
```

App>component.ts

```
import { Component } from '@angular/core';
import{ AuthenticationService } from
'src/app/service/authentication/authentication.service';


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


  constructor(public authenticationService:AuthenticationService) {


  }


logout(){
  this.authenticationService.logout();
}
}
```

App>app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import{ FormsModule, NgModel } from '@angular/forms'
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from './login/login.component';
import {EnableService} from './enable.service';
```

```
import {DisableService} from './disable.service';

import { UserAccountComponent } from './user-account/user-account.component';

import { CheckbookRequestsComponent } from './checkbook-requests/checkbook-
requests.component';

import { HttpClientModule } from '@angular/common/http';

import { AuthorizeRegistrationComponent } from './authorize-
registration/authorize-registration.component';

import { Ng2SearchPipeModule } from 'ng2-search-filter';


import { from } from 'rxjs';



@NgModule({
  declarations: [

    AppComponent,

    LoginComponent,

    UserAccountComponent,

    CheckbookRequestsComponent,

    AuthorizeRegistrationComponent,


  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    HttpClientModule,

    Ng2SearchPipeModule

  ],

  providers: [EnableService,DisableService,],

  bootstrap: [AppComponent]

})

export class AppModule { }
```

app>authorization.service.spec.ts

```
import { TestBed } from '@angular/core/testing';

import { AuthorizationService } from './authorization.service';

describe('AuthorizationService', () => {
  let service: AuthorizationService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(AuthorizationService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { AuthorizeUser } from './model/authorizeUser';

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

  readonly rootUrl = 'http://localhost:8084/user/';

  constructor(private http: HttpClient) { }
```

```
  getRequestData(){
    return this.http.get<AuthorizeUser[]>(this.rootUrl + '/unauthorized/all');
  }

  authorizeAccount(username){
    return this.http.get(this.rootUrl + username + '/authorize');
  }

  rejectRequest(username){
    return this.http.get(this.rootUrl + username + '/authorize/cancel' );

  }

}
import { TestBed } from '@angular/core/testing';

import { CheckbookService } from './checkbook.service';

describe('CheckbookService', () => {
  let service: CheckbookService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(CheckbookService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});
```

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { CheckbookRequest } from './model/checkbookRequest';


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


  //readonly rootUrl =
'localhost:<port>/user/{username}/chequebook/request/confirm';
  readonly rootUrl = 'http://localhost:8084/user/';
  //readonly dataUrl= 'localhost:<port>/chequebook/request/all';
  readonly dataUrl= 'http://localhost:8084/chequebook/request/all';
  private data:any=[]
constructor(private http: HttpClient) { }




confirmCheckbookService(account){
  return this.http.get(this.rootUrl + account +
'/chequebook/request/confirm');
}


getRequestsData():Observable<CheckbookRequest[]> {
  return this.http.get<CheckbookRequest[]>(this.dataUrl);
}


}


import { TestBed } from '@angular/core/testing';
```

```
import { DisableService } from './disable.service';

describe('DisableService', () => {
  let service: DisableService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(DisableService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class DisableService {
  readonly rootUrl = 'http://localhost:8084/user/';

  constructor(private http: HttpClient) {

  }

  disableLoginService(username){
    return this.http.get(this.rootUrl + username + '/disable');
```

```
  }
  }

import { TestBed } from '@angular/core/testing';

import { EnableService } from './enable.service';

describe('EnableService', () => {
  let service: EnableService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(EnableService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

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

  readonly rootUrl = 'http://localhost:8084/user/';
  constructor(private http: HttpClient) {
  }
```

```
  enableLoginService(username){
    return this.http.get(this.rootUrl + username + '/enable');
  }
}

import { TestBed } from '@angular/core/testing';

import { FeaturesService } from './features.service';

describe('FeaturesService', () => {
  let service: FeaturesService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(FeaturesService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

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

```
  id : number
  readonly rootUrl = 'http://localhost:8084/user/';
  constructor(private http: HttpClient) {

  }

  setFeatures(username,value){
    this.id=value
    console.log(this.id)
    return this.http.get(this.rootUrl + username + '/features/' + value);
  }
}

import { TestBed } from '@angular/core/testing';

import { UsersService } from './users.service';

describe('UsersService', () => {
  let service: UsersService;

  beforeEach(() => {
    TestBed.configureTestingModule({});
    service = TestBed.inject(UsersService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
});

import { Injectable } from '@angular/core';
```

```
import { HttpClient } from '@angular/common/http';
import { UserData } from './model/userData';
import { Observable } from 'rxjs';


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


  readonly url = 'http://localhost:8084/';


  constructor(private http:HttpClient) {
   }


   public getAllUsers(){
    return this.http.get<any[]>(this.url+"user/all");
  }
}
```

# Environments

```
export const environment = {
  production: true
};


// This file can be replaced during build by using the `fileReplacements`
array.
// `ng build --prod` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.
```

```
export const environment = {
  production: false
};


/*
 * For easier debugging in development mode, you can import the following file
 * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.
 *
 * This import should be commented out in production mode because it will have a negative impact
 * on performance if an error is thrown.
 */
// import 'zone.js/dist/zone-error';  // Included with Angular CLI.


<!doctype html>
<html lang="en" class="full-height">
<head>
  <meta charset="utf-8">
  <title>TemplateApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.0.4/popper.js"></script>
</head>
<body>
  <app-root></app-root>
</body>
</html>


import { enableProdMode } from '@angular/core';
```

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';


import { AppModule } from './app/app.module';

import { environment } from './environments/environment';


if (environment.production) {

  enableProdMode();

}


platformBrowserDynamic().bootstrapModule(AppModule)

  .catch(err => console.error(err));


/**

 * This file includes polyfills needed by Angular and is loaded before the
app.

 * You can add your own extra polyfills to this file.

 *

 * This file is divided into 2 sections:

 *   1. Browser polyfills. These are applied before loading ZoneJS and are
sorted by browsers.

 *   2. Application imports. Files imported after ZoneJS that should be loaded
before your main

 *      file.

 *

 * The current setup is for so-called "evergreen" browsers; the last versions
of browsers that

 * automatically update themselves. This includes Safari >= 10, Chrome >= 55
(including Opera),

 * Edge >= 13 on the desktop, and iOS 10 and Chrome on mobile.

 *

 * Learn more in https://angular.io/guide/browser-support

 */
```

```
/***************************************************************************
********************

 * BROWSER POLYFILLS

 */


/** IE10 and IE11 requires the following for NgClass support on SVG elements
*/

// import 'classlist.js';  // Run `npm install --save classlist.js`.


/**

 * Web Animations `@angular/platform-browser/animations`

 * Only required if AnimationBuilder is used within the application and using
IE/Edge or Safari.

 * Standard animation support in Angular DOES NOT require any polyfills (as of
Angular 6.0).

 */

// import 'web-animations-js';  // Run `npm install --save web-animations-js`.


/**

 * By default, zone.js will patch all possible macroTask and DomEvents

 * user can disable parts of macroTask/DomEvents patch by setting following
flags

 * because those flags need to be set before `zone.js` being loaded, and
webpack

 * will put import in the top of bundle, so user need to create a separate
file

 * in this directory (for example: zone-flags.ts), and put the following flags

 * into that file, and then add the following code before importing zone.js.

 * import './zone-flags';

 *

 * The flags allowed in zone-flags.ts are listed here.

 *

 * The following flags will work for all browsers.

 *
```

```
 * (window as any).__Zone_disable_requestAnimationFrame = true; // disable
patch requestAnimationFrame

 * (window as any).__Zone_disable_on_property = true; // disable patch
onProperty such as onclick

 * (window as any).__zone_symbol__UNPATCHED_EVENTS = ['scroll', 'mousemove'];
// disable patch specified eventNames

 *

 *  in IE/Edge developer tools, the addEventListener will also be wrapped by
zone.js

 *  with the following flag, it will bypass `zone.js` patch for IE/Edge

 *

 *  (window as any).__Zone_enable_cross_context_check = true;

 *

 */


/***************************************************************************
*********************

 * Zone JS is required by default for Angular itself.

 */

import 'zone.js/dist/zone';  // Included with Angular CLI.


/***************************************************************************
*********************
 * APPLICATION IMPORTS

 */


// This file is required by karma.conf.js and loads recursively all the .spec
and framework files


import 'zone.js/dist/zone-testing';

import { getTestBed } from '@angular/core/testing';

import {

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting
```

```
} from '@angular/platform-browser-dynamic/testing';


declare const require: {
  context(path: string, deep?: boolean, filter?: RegExp): {
    keys(): string[];
    <T>(id: string): T;
  };
};


// First, initialize the Angular testing environment.
getTestBed().initTestEnvironment(
  BrowserDynamicTestingModule,
  platformBrowserDynamicTesting()
);
// Then we find all the tests.
const context = require.context('./', true, /\.spec\.ts$/);
// And load the modules.
context.keys().map(context);
```