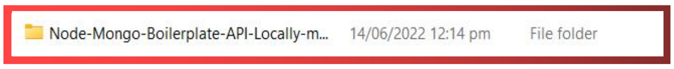
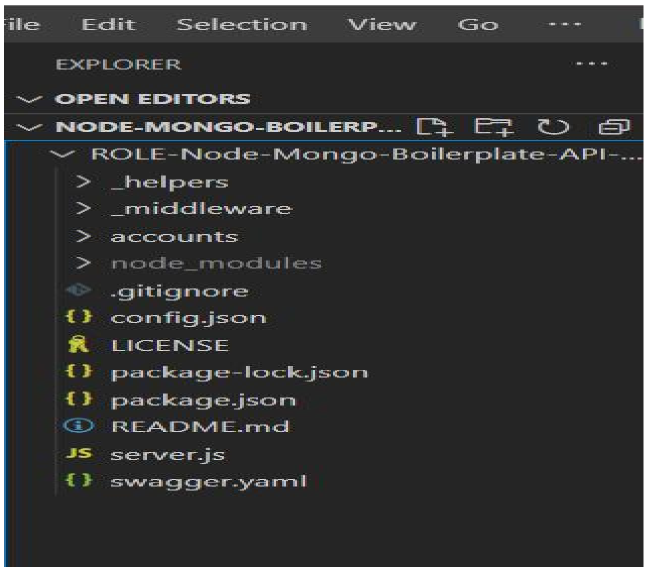
USER MANAGEMENT WITH ROLES

1. Clone Node-Mongo-Boilerplate-API-Locally in Github



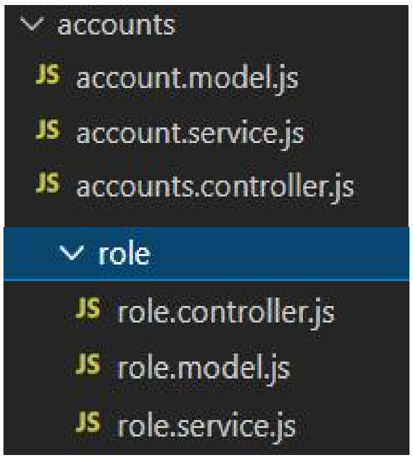
```
Node-Mongo-Boilerplate-API-Locally-m...   14/06/2022 12:14 pm   File folder
```

2. Open Visual studio code and drag the folder



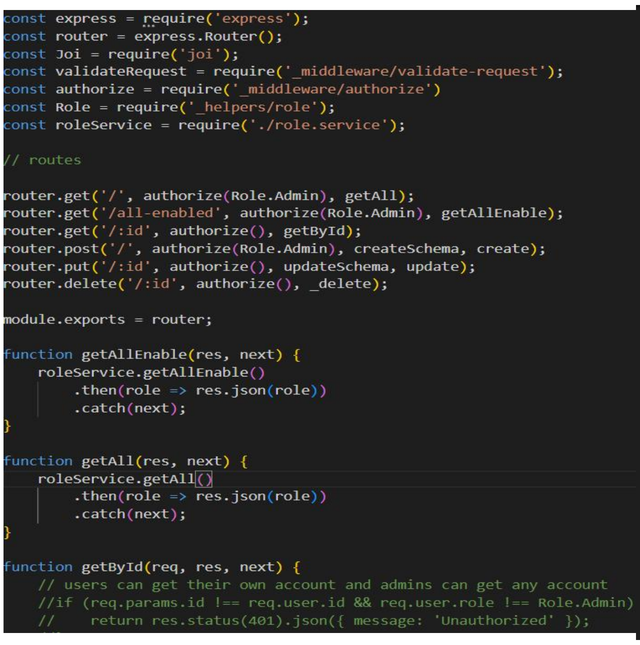
```
NODE-MONGO-BOILERP...
  ROLE-Node-Mongo-Boilerplate-API-...
    > _helpers
    > _middleware
    > accounts
    > node_modules
    .gitignore
    config.json
    LICENSE
    package-lock.json
    package.json
    README.md
    server.js
    swagger.yaml
```

3. Open Terminal and type npm install
4. Copy the account file renamed it role and Replace the account name into " role"

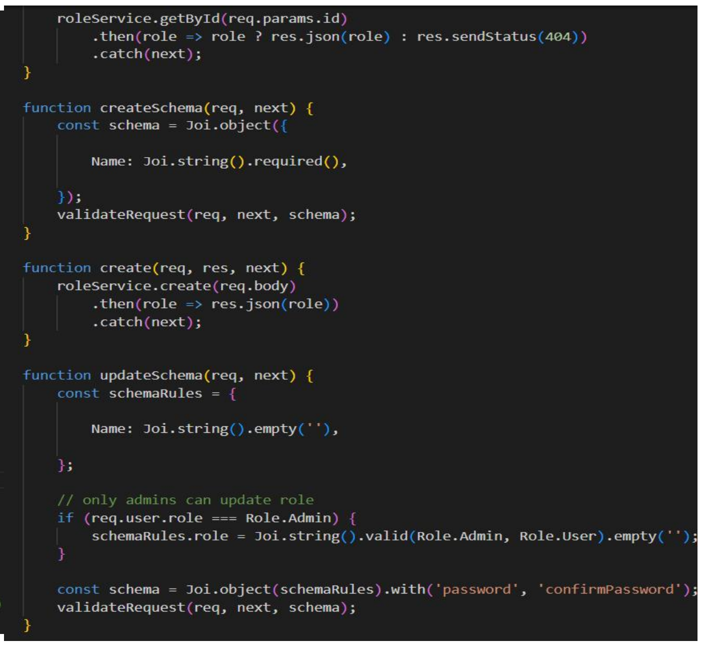


```
accounts
  account.model.js
  account.service.js
  accounts.controller.js
  role
    role.controller.js
    role.model.js
    role.service.js
```

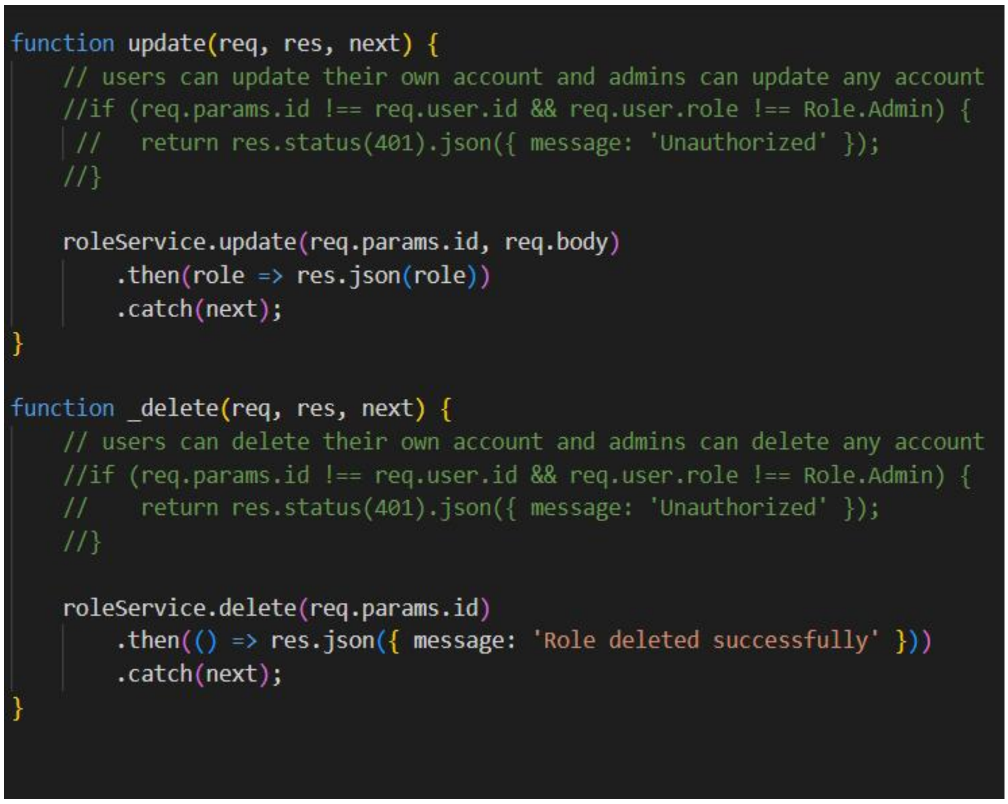
JS role.controller.js



```javascript
const express = require('express');
const router = express.Router();
const Joi = require('joi');
const validateRequest = require('_middleware/validate-request');
const authorize = require('_middleware/authorize')
const Role = require('_helpers/role');
const roleService = require('./role.service');

// routes

router.get('/', authorize(Role.Admin), getAll);
router.get('/all-enabled', authorize(Role.Admin), getAllEnable);
router.get('/:id', authorize(), getById);
router.post('/', authorize(Role.Admin), createSchema, create);
router.put('/:id', authorize(), updateSchema, update);
router.delete('/:id', authorize(), _delete);

module.exports = router;

function getAllEnable(res, next) {
    roleService.getAllEnable()
        .then(role => res.json(role))
        .catch(next);
}

function getAll(res, next) {
    roleService.getAll()
        .then(role => res.json(role))
        .catch(next);
}

function getById(req, res, next) {
    // users can get their own account and admins can get any account
    //if (req.params.id !== req.user.id && req.user.role !== Role.Admin)
    //    return res.status(401).json({ message: 'Unauthorized' });
```

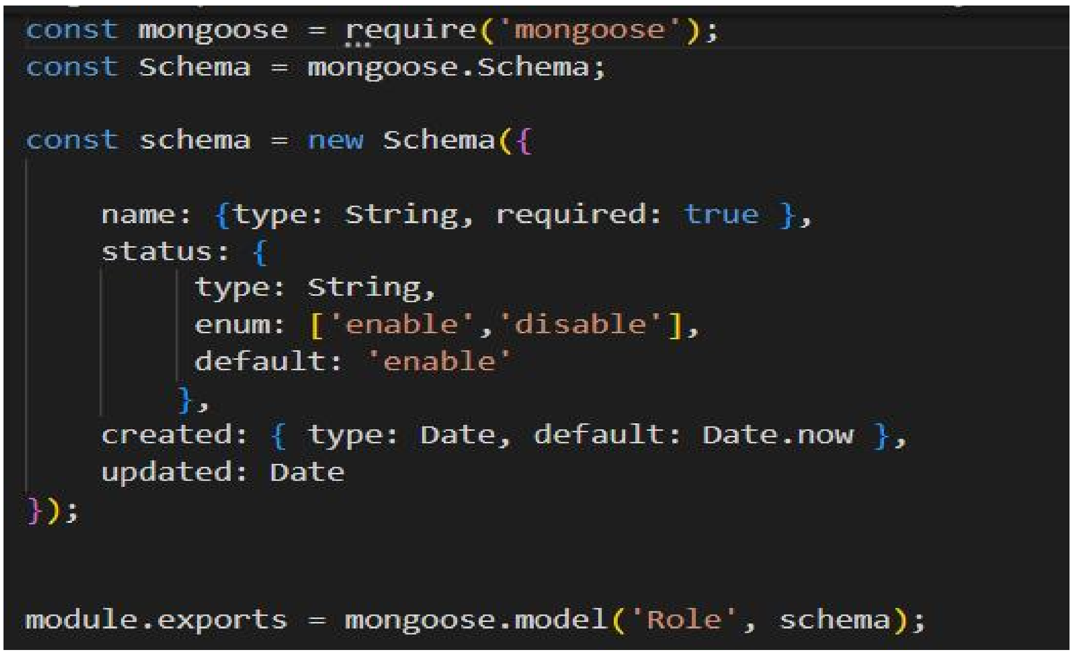


```javascript
    roleService.getById(req.params.id)
        .then(role => role ? res.json(role) : res.sendStatus(404))
        .catch(next);
}

function createSchema(req, next) {
    const schema = Joi.object({

        Name: Joi.string().required(),

    });
    validateRequest(req, next, schema);
}

function create(req, res, next) {
    roleService.create(req.body)
        .then(role => res.json(role))
        .catch(next);
}

function updateSchema(req, next) {
    const schemaRules = {

        Name: Joi.string().empty(''),

    };

    // only admins can update role
    if (req.user.role === Role.Admin) {
        schemaRules.role = Joi.string().valid(Role.Admin, Role.User).empty('');
    }

    const schema = Joi.object(schemaRules).with('password', 'confirmPassword');
    validateRequest(req, next, schema);
}
```



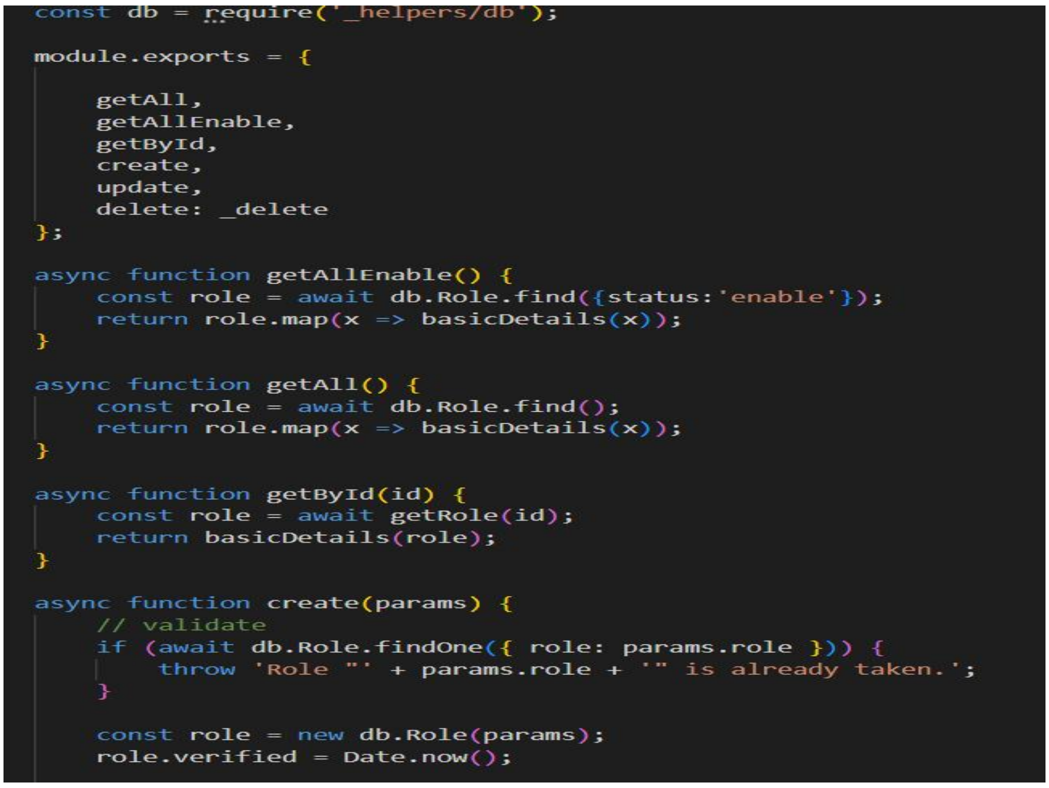
```javascript
function update(req, res, next) {
    // users can update their own account and admins can update any account
    //if (req.params.id !== req.user.id && req.user.role !== Role.Admin) {
    //    return res.status(401).json({ message: 'Unauthorized' });
    //}

    roleService.update(req.params.id, req.body)
        .then(role => res.json(role))
        .catch(next);
}

function _delete(req, res, next) {
    // users can delete their own account and admins can delete any account
    //if (req.params.id !== req.user.id && req.user.role !== Role.Admin) {
    //    return res.status(401).json({ message: 'Unauthorized' });
    //}

    roleService.delete(req.params.id)
        .then(() => res.json({ message: 'Role deleted successfully' }))
        .catch(next);
}
```

JS role.model.js



```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;

const schema = new Schema({

    name: {type: String, required: true },
    status: {
         type: String,
         enum: ['enable','disable'],
         default: 'enable'
        },
    created: { type: Date, default: Date.now },
    updated: Date
});


module.exports = mongoose.model('Role', schema);
```

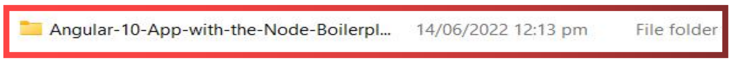
JS role.service.js



```
const db = require('_helpers/db');

module.exports = {

    getAll,
    getAllEnable,
    getById,
    create,
    update,
    delete: _delete
};

async function getAllEnable() {
    const role = await db.Role.find({status:'enable'});
    return role.map(x => basicDetails(x));
}

async function getAll() {
    const role = await db.Role.find();
    return role.map(x => basicDetails(x));
}

async function getById(id) {
    const role = await getRole(id);
    return basicDetails(role);
}

async function create(params) {
    // validate
    if (await db.Role.findOne({ role: params.role })) {
        throw 'Role "' + params.role + '" is already taken.';
    }

    const role = new db.Role(params);
    role.verified = Date.now();
```

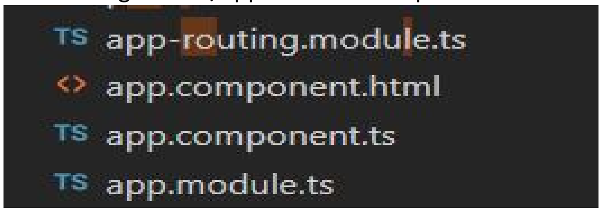
Angular-10-App-with-the-Node-Boilerplate-API-master

1. Clone Angular-10-App-with-the-Node-Boilerplate-API-master in Github



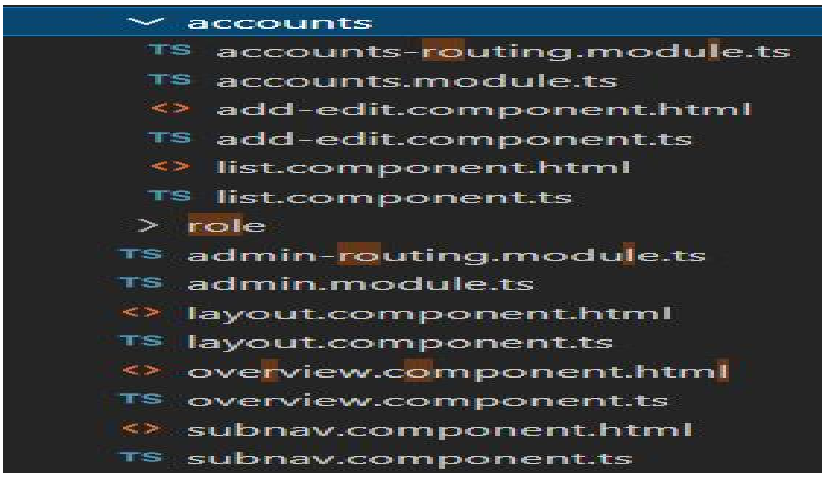
```
Angular-10-App-with-the-Node-Boilerpl...    14/06/2022 12:13 pm    File folder
```

2. Open you Visual Studio Code drag your folder to vs code open terminal and type "npm install" after that go to scr/appctrl + f or find profile



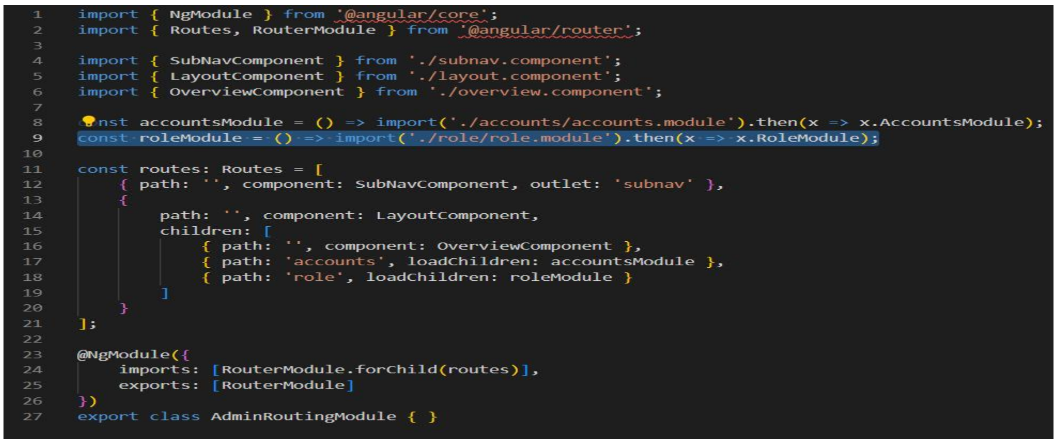
```
app-routing.module.ts
app.component.html
app.component.ts
app.module.ts
```

3. Add folder roles in the admin and copy the same from accounts to your role folder
You can edit the files and code and make changes in your account role



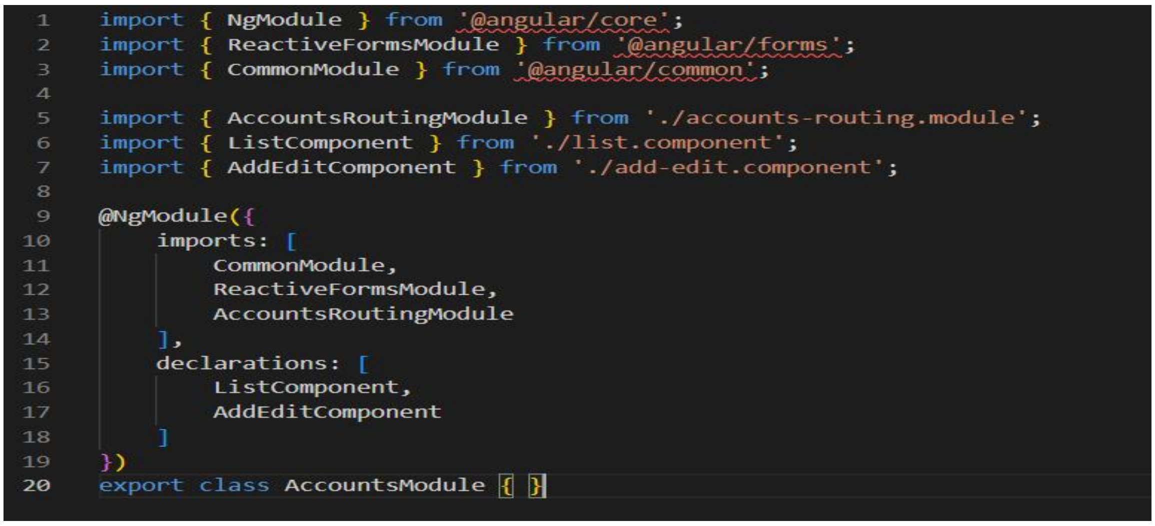
```
accounts
    accounts-routing.module.ts
    accounts.module.ts
    add-edit.component.html
    add-edit.component.ts
    list.component.html
    list.component.ts
  role
  admin-routing.module.ts
  admin.module.ts
  layout.component.html
  layout.component.ts
  overview.component.html
  overview.component.ts
  subnav.component.html
  subnav.component.ts
```

4.Find admin-routing.module.ts and add the code " const roleModule = () => import('./role/role.module').then(x => x.RoleModule);   "



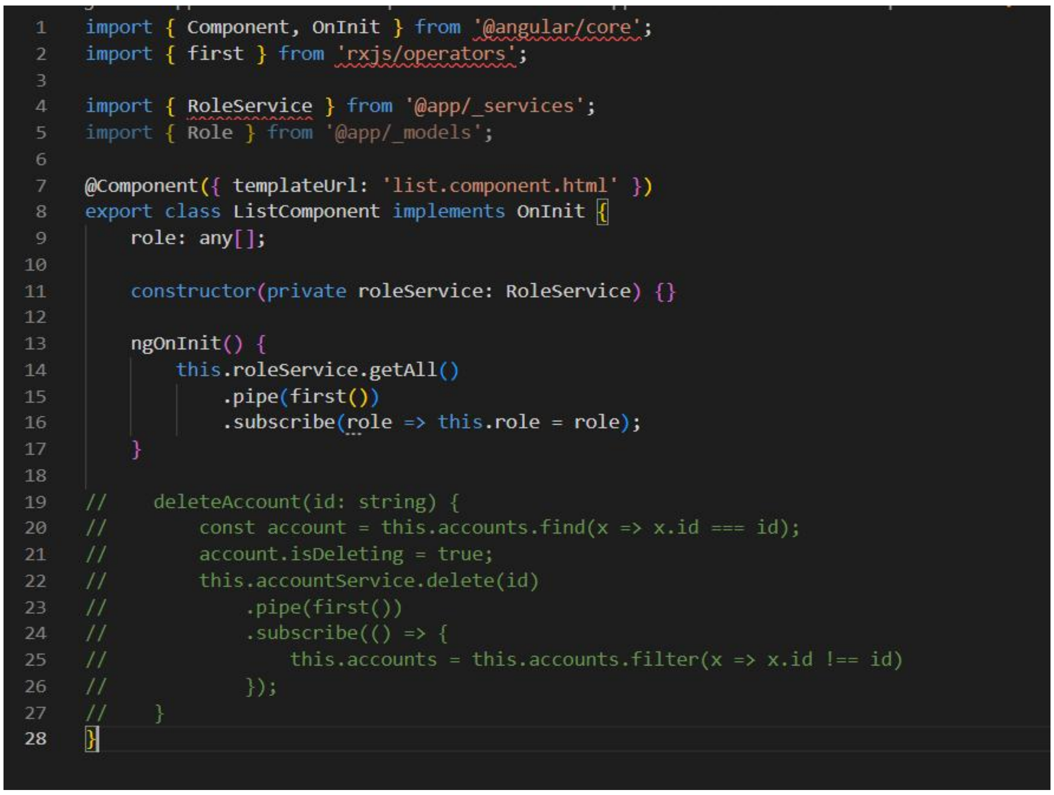
```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';

import { SubNavComponent } from './subnav.component';
import { LayoutComponent } from './layout.component';
import { OverviewComponent } from './overview.component';

nst accountsModule = () => import('./accounts/accounts.module').then(x => x.AccountsModule);
const roleModule = () => import('./role/role.module').then(x => x.RoleModule);

const routes: Routes = [
    { path: '', component: SubNavComponent, outlet: 'subnav' },
    {
        path: '', component: LayoutComponent,
        children: [
            { path: '', component: OverviewComponent },
            { path: 'accounts', loadChildren: accountsModule },
            { path: 'role', loadChildren: roleModule }
        ]
    }
];

@NgModule({
    imports: [RouterModule.forChild(routes)],
    exports: [RouterModule]
})
export class AdminRoutingModule { }
```

5. role.routing.module.ts



```typescript
import { NgModule } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';

import { AccountsRoutingModule } from './accounts-routing.module';
import { ListComponent } from './list.component';
import { AddEditComponent } from './add-edit.component';

@NgModule({
    imports: [
        CommonModule,
        ReactiveFormsModule,
        AccountsRoutingModule
    ],
    declarations: [
        ListComponent,
        AddEditComponent
    ]
})
export class AccountsModule { }
```

6. List.component.ts



```typescript
import { Component, OnInit } from '@angular/core';
import { first } from 'rxjs/operators';

import { RoleService } from '@app/_services';
import { Role } from '@app/_models';

@Component({ templateUrl: 'list.component.html' })
export class ListComponent implements OnInit {
    role: any[];

    constructor(private roleService: RoleService) {}

    ngOnInit() {
        this.roleService.getAll()
            .pipe(first())
            .subscribe(role => this.role = role);
    }

//    deleteAccount(id: string) {
//        const account = this.accounts.find(x => x.id === id);
//        account.isDeleting = true;
//        this.accountService.delete(id)
//            .pipe(first())
//            .subscribe(() => {
//                this.accounts = this.accounts.filter(x => x.id !== id)
//            });
//    }
}
```