1.) at first create component :-- header,footer,home,login,pagenot found

2) add bootstrap cdn link

3) design header,footer,page not found component

4) add routing path for all component in **routing.module.ts** file

5) add **routerlink** in all nav bar, and add class **routerLinkActive="Active"** for highlighting active nav bar

6) crete one lazy-loading module with the help of **ng g m modules/todo --route todo --module app.module**

7) goto **todo component.html** page and degisn this page, at first break this page 2 colum wise,,

8) create service for api using **ng g s modules/service/todo-api**

9) goto **todo-module.ts** file and **app.module.ts** file and import

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

10) then go to service and import httpclient, and create a methode name:- **getAllTodos()**

11.a) go to **todo-component.ts** and import services

create one model name is todo.ts

here call **getAllTodo()** and subscribe this

11.b) create one methode name:- **Populatetodos()** and under this methode we call **getAllTodo()** and subscribe this methode

11.c) and call **Populatetodos()** into the **ngOnInit.**

11.D) DISPLAY ALL DATA INTO TODO.COMPONENT.HTML PAGE and create spinner component under todo module

12.a) run **npm i ngx-pagination** and import **NgxPaginationModule** in todo.module.ts

12.b) then go to html page and set pagination

12.c) then install **npm i ngx-pagination** and create search option

13.a) create todo operation part (====== POST METHODE==== .............................................................................................

13.b) now create post methode for create new todo..

1st go to ts file and create a public data member whose type is modal name.

then go to html file and under todos operation portion we bind all element using ngModel ( here use two way data binding)

<div class="form-group">

<label>Title:</label>

<input type="text" name="title" class="form-control" placeholder="Please Enter The Title" [(ngModel)]="newTodo.title">

</div> // [(ngModel)]="newTodo.title ( this "title" name should be same as model name)

<div class="form-group">

<label>Description:</label>

<textarea name="desc" id="" cols="10" rows="3" class="form-control" placeholder="Please Enter Description " [(ngModel)]="newTodo.desc"></textarea>

</div>

13.c) go to ts file

under constructor we create new todo by using new keyword

constructor(private apiservices: TodoApiService) {

this.newTodo = new todo('meanstack','this is mongodb','this.apiservices.getdate()');

}

13.d) create **custom date method** into services for getting current time

13.e) create addTodo() function into todo.component.ts file

// create new todo function

AddTodo(){

console.log(this.newTodo);

this.apiservices.addTodo(this.newTodo).subscribe((item:any)=>{

console.log(item);

alert(item.msg);

this.populateTode();

});

}

14.A) then create radion button inside todos operation and create one change methode name select(a). after this go to **todo.component.ts** file and create select(a) function

//select todo from todo's operation

public selectedTodo: any = [];

select(a: any) {

this.selectedTodo = a;

//console.log(this.selectedTodo);

this.newTodo = new todo(this.selectedTodo.title, this.selectedTodo.desccription, this.apiservices.getDate());

}

14.b) now goto services and crete updateTodo() methode

**/// selected todo update , put methode**

updateTodo(newTodo:todo,id:any):Observable<todo>{

return this.http.put<todo>(' https://todolist-api.glitch.me/api/todo/'+id,newTodo);

}

14.c) go to ts file and create

**// for update todo, put method()**

updateTodo(){

console.log(this.newTodo);

this.apiservices.updateTodo(this.newTodo,this.selectedTodo.\_id).subscribe((item:any)=>{

//console.log(item);

this.populateTode();

alert(item.msg);

});

}

15.a) design login page and sign in page

15.b) create 2 model under user folder , name is:- login.ts, signup.ts

15.c) import formModule in app.module.ts

15.d) go to **login component.ts** file

export class LoginComponent implements OnInit {

// for login

  public loguser:any=[];

  constructor() {

    this.loguser= new login('rupams441@gmail.com','aA1@aaaa')

  }

  ngOnInit(): void {

  }

}

Go to html file and bile with ngModel

Like:-- name="email"

[(ngModel)]="loguser.email"

And:-- name="pas1"

[(ngModel)]="loguser.pas1"

// name should be same as modal name, and don’t use normal from tag

15.e) above same work do in our signin cmponent

Using template driven from of signup page, use reactive form of login page

8) .if you want use json server insted of api then use bellow code

.............................................start backend work...........................................

8.a) at first create a folder name server (outside ther src folder) then right click this folder name and click "open in intrigrity terminal"

then run a command in cmd ( npm init --yes) then we write "npm install json-server"

8.b) create db.json file under server folder

8.c) package.json file configuration

go to package.json file under server folder

under script portion delete "test" column and write

"start": "json-server --watch db.json -p 4000"

8.d) run db.json file through "npm start" command

15.f) create a service under Loginservice folder

**ng g s LoginService/loginService**

 constructor(private http:HttpClient) { }

  // user signup

  signup(userdata:any){

    return this.http.post('http://localhost:4000/signup', userdata);

  }

  // login

  login(logindata:any){

    return this.http.post('http://localhost:4000/login',logindata);

  }

**go to login service.ts file**

// for login

  public loguser:any=[];

  login(){

  //  console.log(this.loguser);

   this.loginapi.login(this.loguser).subscribe((item:any)=>{

     console.log(item);

   alert('login successfully');

   this.route.navigateByUrl('/home');

   }, err=>{

    alert('sorry!!!,please try again')

   });

  }

**Go to signup.ts file**

SignUp(){

  console.log(this.newuser);

  let userdata=this.newuser;

  this.Loginapi.signup(userdata).subscribe((res:any)=>{

    console.log(res);

    // alert(res.message)

    alert('SignUp Successfully');

    this.route.navigateByUrl('/login');

  },err=> {

    alert('sorry!!!,please try again');

  }

  );

  }

**GOTO LOGIN.COMPONENT.TS**

login(){

  this.http.get<any>('http://localhost:4000/signup').subscribe((res:any)=>{

    const user = res.find((a:any)=>{

      return a.email === this.loguser.email && a.pas1 === this.loguser.pas1

    })

    if(user){

      alert('Login successfull')

      this.route.navigateByUrl('/todo')

    }

    else{

      alert('User Not Found!!! please go to Signup Page....')

    }

  })

  }

17)now we create gurd under modules folder

**ng g g modules/guard/auth**

**and select canActivate**

**at first go to login.component.ts file**

// for login

  login(){

  this.http.get<any>('http://localhost:4000/signup').subscribe((res:any)=>{

    const user = res.find((a:any)=>{

      return a.email === this.loguser.email && a.pas1 === this.loguser.pas1

    })

    if(user){

      alert('Login successfull')

      localStorage.setItem('email',this.loguser.email); // here set email into our local storage

      localStorage.setItem('password',this.loguser.pas1); // here set password into our local storage

      this.route.navigateByUrl('/todo')

    }

    else{

      alert('User Not Found!!! please go to Signup Page....')

    }

  })

  }

Then go to auth.guard.ts file:

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

import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree } from '@angular/router';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class AuthGuard implements CanActivate {

  canActivate(): boolean{

    let token=localStorage.getItem('password');

    if(!token){

      alert('Please login to view this page ....!');

      return false;

    }

    return true;

  }

}

Then go to app.routing.module.ts file

const routes: Routes = [

  {path:'',component:HomeComponent,pathMatch:'full'},

  {path:'home',component:HomeComponent},

  {path:'login',component:LoginComponent},

  {path:'signup',component:SignupComponent},

  { path: 'todo', loadChildren: () => import('./modules/todo/todo.module').then(m => m.TodoModule),canActivate:[AuthGuard] }, // here set canActive authguard

  {path:'\*\*', component:PageNotFoundComponent}

];

Project help:-- project video of ejob and

https://youtu.be/opHac35xM5U

In this project we use json server for login and logout

Project name:-- TodoList-2023

Todolist

1. create a service under Loginservice folder

if you use original api then use bellow code

**ng g s LoginService/loginService**

constructor(private http:HttpClient) { }

    //User signUp for api:

    signUp(userdata:any){

      return this.http.post('https://rest-food-api.glitch.me/api/user/signup',userdata);

     }

    // user Login for api :

     login(logIndata:any){

      return this.http.post('https://rest-food-api.glitch.me/api/user/login',logIndata);

     }

}

1. Go to login.component.ts

  login() {

    //console.log(this.loguser);

    // let logIndata = this.loguser;

    this.loginapi.login(this.loguser).subscribe((res: any) => {

      console.log(res);

      alert('LogIn Successfully....!');

      if (res.message == 'success') {

        localStorage.setItem('User', res.loggedUser);

        localStorage.setItem('token', res.token);

      }

      this.route.navigateByUrl('/todo')

    });

  }

1. Go to sgnup.component.ts
2. We create gurd name auth:

**ng g g modules/guard/auth**

**and select canActivate**

1. import { Injectable } from '@angular/core';

import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, UrlTree } from '@angular/router';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class AuthGuard implements CanActivate {

  canActivate(): boolean{

    let token=localStorage.getItem('token');

    if(!token){

      alert('Please login to view this page ....!');

      return false;

    }

    return true;

  }

}

https://rupam-todolist.000webhostapp.com/home

**Todolist Rest API Documentation**

**BASE URL:** https://todolist-api.glitch.me/

1)Getting all Todos from API

REQUEST : GET

URL : https://todolist-api.glitch.me/api/todos

Params :None.

2)Getting Perticular Todo depends on Id from API

REQUEST: GET

URL : https://todolist-api.glitch.me/api/todo/{id}

Params :None

3)Adding a new Todo into database through API

REQUEST : POST

URL : https://todolist-api.glitch.me/api/todo

Params : title,desc,created

4)Updating exisiting Todo depends on ID

REQUEST: PUT

URL : https://todolist-api.glitch.me/api/todo/{id}

Params : title,desc,created.

5)Deleting Todo depends on id

REQUEST : DELETE

URL : https://todolist-api.glitch.me/api/todo/{id}

Params : None.

**UserSignUp /SignIn Api Documentation:**

--------------------------------------

User module

SIGNUP :

REQUEST: POST

URL :https://rest-food-api.glitch.me/api/user/signup

Params :uname,email,pass1 [string]

SIGNIN/LOGIN :

Request : POST

URL :https://rest-food-api.glitch.me/api/user/login

Params :email,pass1 [string].