Functional And Unit Testing

- The application consists of :-

1)Login page:-

The page contains two blank spaces that are to be filled with the username and password in order to log into the website. Inside the class of the component, there is an 'ngOnInit' method that initializes the component. It creates an instance of 'FormGroup' using 'formBuilder. group' and assigns it to the 'form' property of the component. The 'FormGroup' represents the form in the component and holds the form controls. The component also has a 'submit' method that is called when a form submission occurs. Inside the method, an HTTP POST request is made using http.post to a specific URL (http://localhost:8000/api//login).The request payload is obtained from the 'form.get RawValue()'method, which retrieves the form's raw values. The option withCredentials: true indicates that the request should include credentials (such as cookies) when sent.If the POST request is successful, the subscribe method is called, and upon receiving the response, the router.navigate(['/']) method is invoked to navigate to the root URL (in this case, '/').Docker is deployed for authentication that created containers.

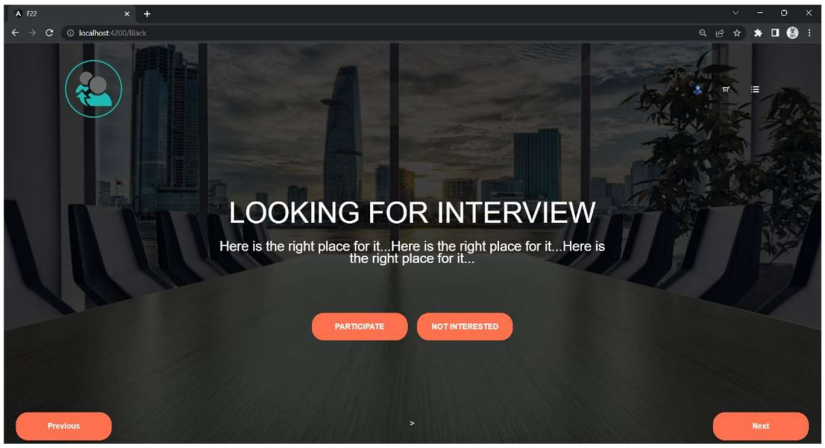
2)Sign up page:-

Before Loging in, the user has the register his or her details in the sign up page, the data that is entered will be stored in the containers created by docker which is then used to log into the main page. Inside the class of the component there is an ngOnInit method that initializes the component. It creates an instance of FormGroup using formBuilder. Group and assigns it to the form property of the component. The FormGroup represents the form in the component and holds the form controls. The component also has a submit method that is called when a form submission occurs. Inside the method, an HTTP POST request is made using http.post to a specific URL(**http://localhost:8000/api/register**). The request payload is obtained from the form.getRawValue() method, which retrieves the form's raw values. The POST request sends the

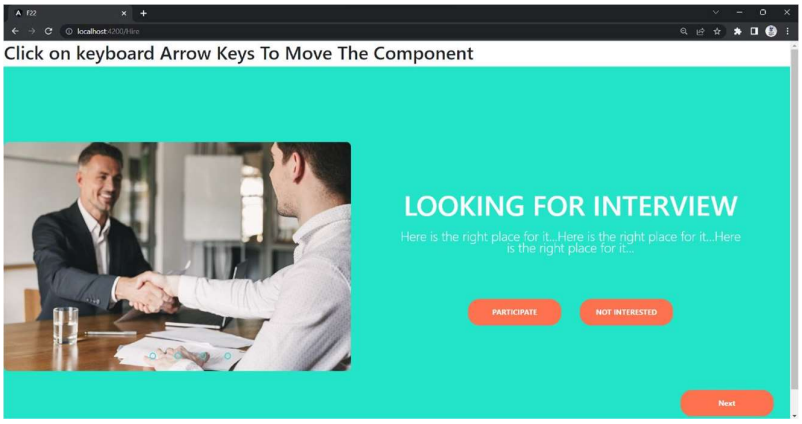
form data to the specified URL for user registration. If the POST request is successful, the subscribe method is called, and upon receiving the response, the router.navigate(['/login']) method is invoked to navigate to the '/login' route, typically the login page. In summary, the code sets up a registration form in the RegComponent using FormGroup and FormBuilder from @angular/forms. It handles form submission by making an HTTP POST request to a registration API endpoint and redirects to the login page after successful registration.

3)Main Components:-

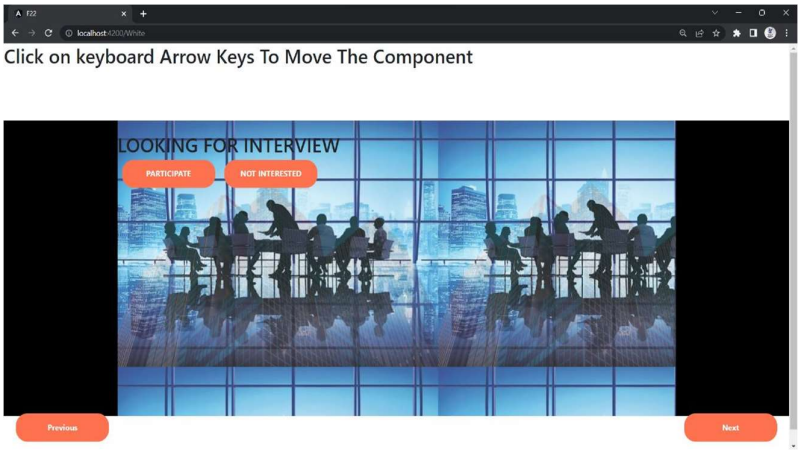
i)Hiring Component:-



ii)Job Component:-

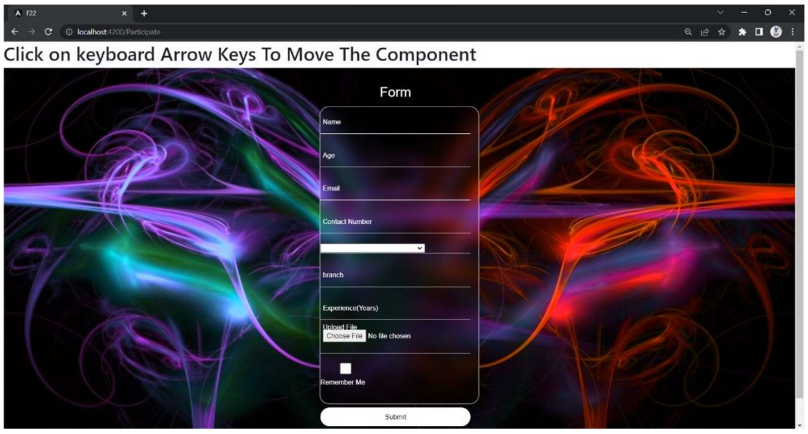


iii)Walkin Component:-



Inside the classes of each component, there is a constructor that injects an instance of the Router class. This is done through dependency injection, allowing the component to use the Router for navigation. The ngOnInit method is empty in this code, so it does not contain any specific logic. The ngOnInit lifecycle hook is typically used to perform initialization tasks when the component is being initialized. The component also defines two methods:
openParticipate and openEnd.
These methods are called when certain actions or events occur, such as button clicks or links being clicked in the component's template.
openParticipate method: This method uses the this.router.navigate([‘/Participate’]) statement to navigate to the '/Participate'route.openEnd method: This method uses the this.router.navigate(['/End']) statement to navigate to the '/End' route.

4)Form Component:-



Inside the ParticipateComponent class, there are several properties and methods:-

1) form property:

This property holds an instance of FormGroup created using the new FormGroup method. It represents the form in the component and holds the form controls.

2)profile property:

This property represents a user profile object of type profile.

3)imageData property:

This property stores the base64-encoded data of the selected image file.

4)loading property:

This property indicates whether the form submission is in progress or not.

5)rememberMeChecked property:

This property represents the state of the "Remember Me" checkbox in the form. The ngOnInit method is called when the component is being

initialized. Inside this method, the form property is initialized by creating an instance of FormGroup using new FormGroup and specifying

form controls with their initial values and validators.

The component also defines several other methods:-

1)onFileSelect method:

This method is called when a file is selected using an input field. It extracts the selected file, updates the image form control's value with the file, and converts the file to base64 to display a preview of the image.

2)onSubmit method:

This method is called when the form is submitted. It checks if the form is invalid and displays an error message if any required fields are missing. If the form is valid, it prepares form data using FormData, adds the form values to the data, and passes it to the profileService .addProfile method to add the profile. If the "Remember Me" checkbox is checked, the form values are stored in the local storage. Finally, after a brief delay, the loading state is reset, the form is cleared, and the user is navigated to '/next-page'.

3)onRememberMeChange method:

This method is called when the "Remember Me" checkbox state changes. It updates the rememberMeChecked property based on the checkbox's value.

4)loadRememberedCredentials method:

This method loads the user credentials from the local storage and populates the form fields if the credentials are available. It

also updates the rememberMeChecked property accordingly.

5)Home Component:-

Inside the HomeComponent class, there are several properties and methods:-

1)activePage property:

This property holds the active page enum value, which represents the currently active page among JOB, WALKIN, and HIRE.

2)constructor method:

This method is the component's constructor and injects an instance of the Router class using dependency injection.

3)ngOnInit method:

This method is called when the component is being initialized. It subscribes to the router.events observable to listen for navigation events. When a NavigationEnd event occurs, it checks the current URL and sets the activePage property accordingly.

4)ngOnDestroy method:

This method is called when the component is being destroyed. It unsubscribes from the router.events observable to prevent memory leaks.

5)@HostListener('document:keydown', ['$event'])decorator and handleKeyboardEvent method:

This combination allows the component to listen to keyboard events. Specifically, it listens for the 'ArrowLeft' and 'ArrowRight' keys. If the 'ArrowLeft' key is pressed, the previousPage method is called, and if the 'ArrowRight' key is pressed, the nextPage method is called.

6)previousPage method:

This method navigates to the previous page based on the current activePage value. It uses the router.navigateByUrl method router.navigateByUrl method to navigate to the corresponding URL.

7)nextPage method:

This method navigates to the next page based on the current activePage value. It uses the router.navigateByUrl method to navigate to the corresponding URL.

8)showPage1, showPage2, and showPage3 methods:

These methods navigate to specific pages (/White, /Black, and /Hire, respectively) using the router.navigateByUrl method.