

FORMS

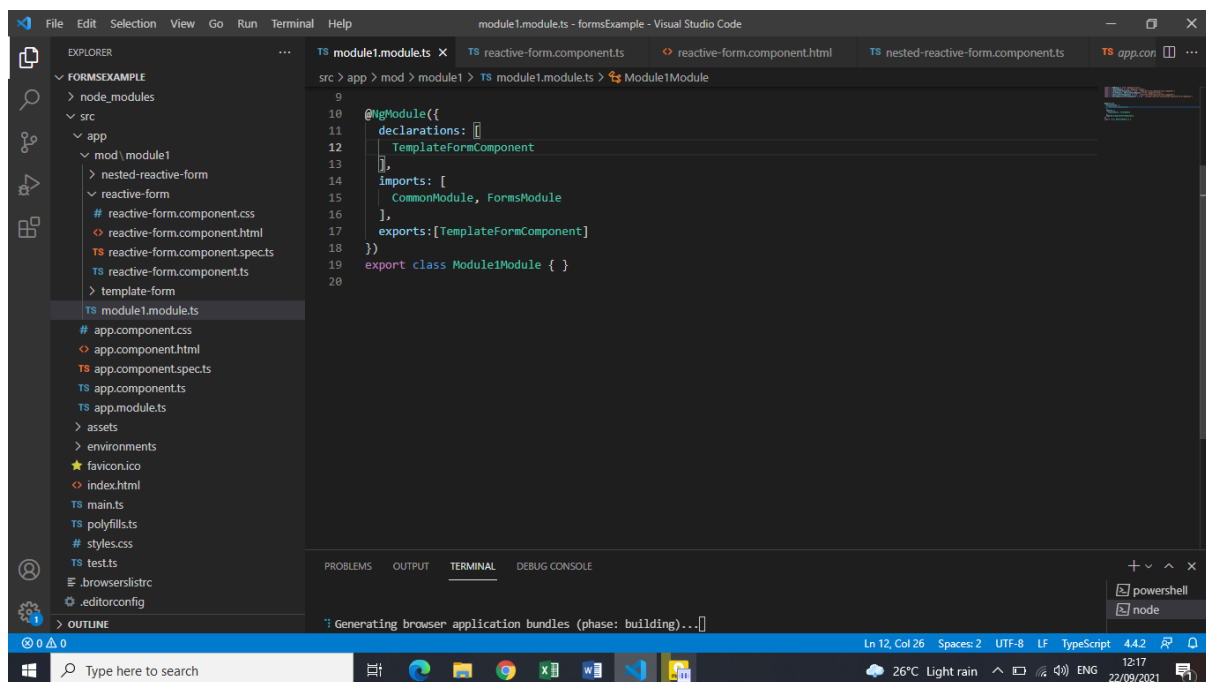
Handling user input with forms is the cornerstone of many common applications. Applications use forms to enable users to log in, to update a profile, to enter sensitive information, and to perform many other data-entry tasks.

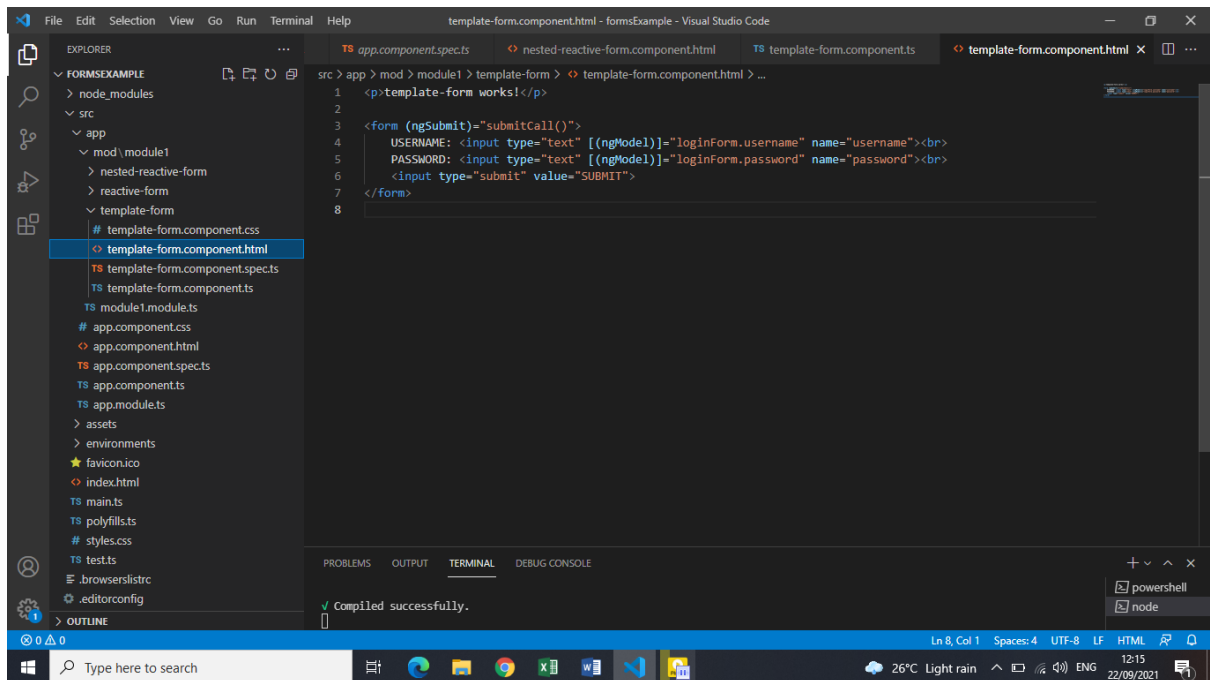
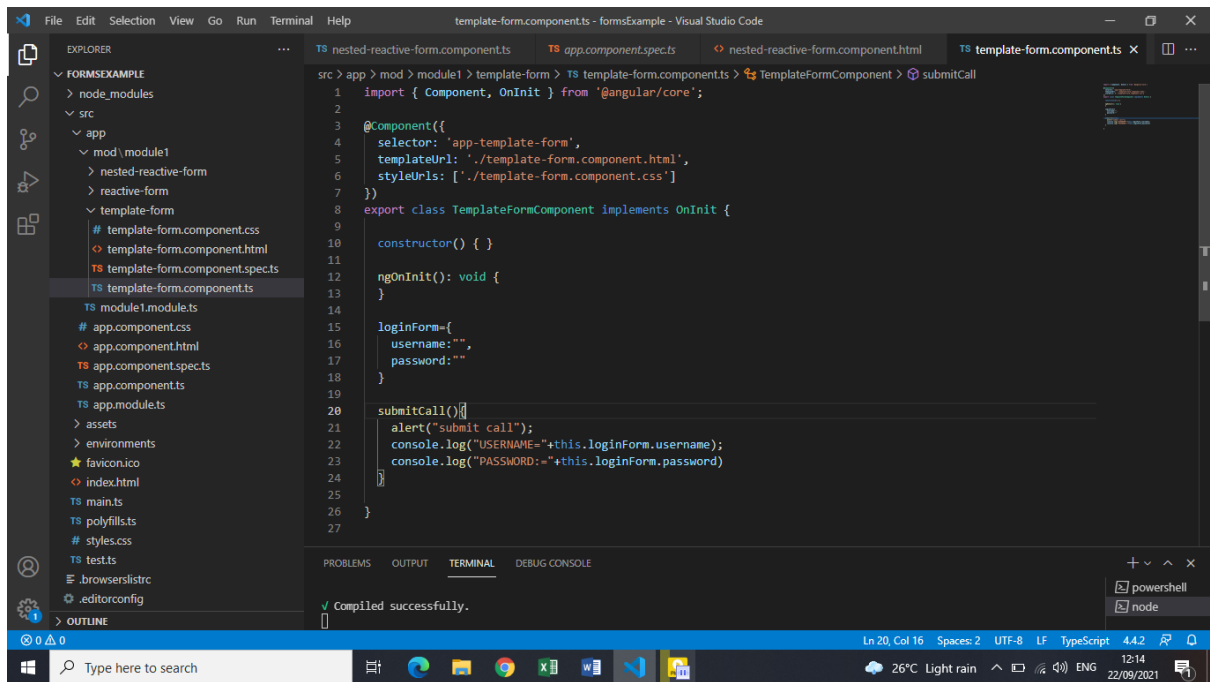
Angular provides two different approaches to handling user input through forms: reactive and template-driven. Both capture user input events from the view, validate the user input, create a form model and data model to update, and provide a way to track changes.

This guide provides information to help you decide which type of form works best for your situation. It introduces the common building blocks used by both approaches. It also summarizes the key differences between the two approaches, and demonstrates those differences in the context of setup, data flow, and testing.

1) Template-driven forms:

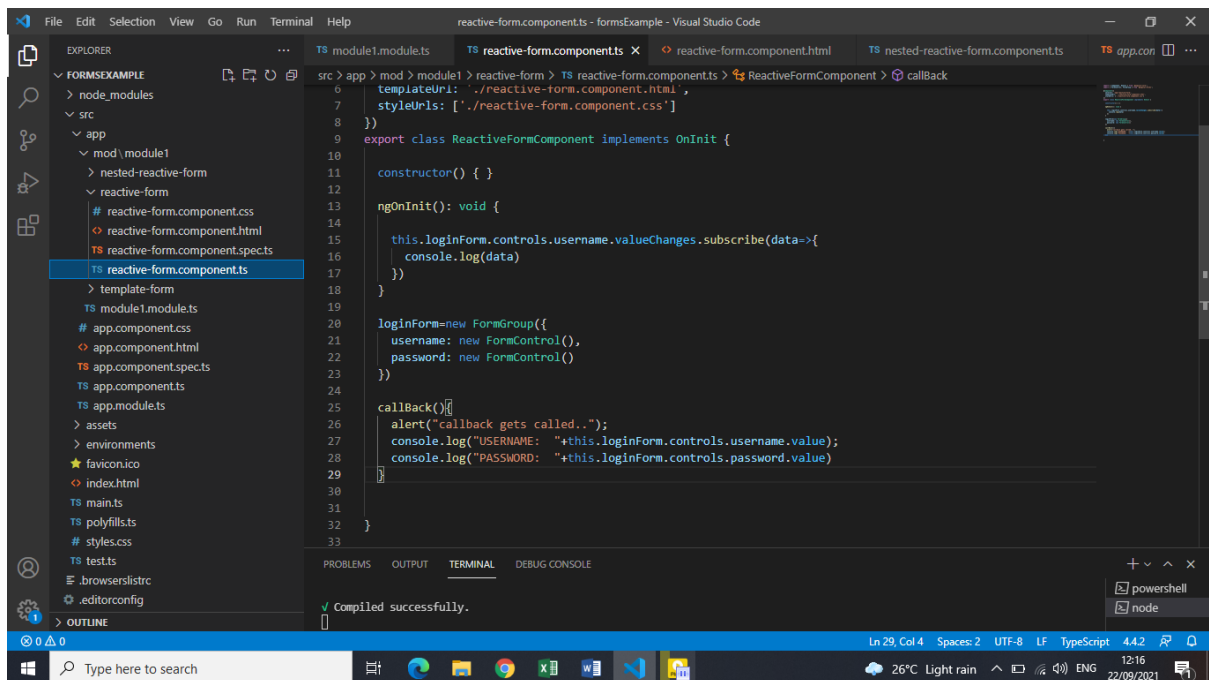
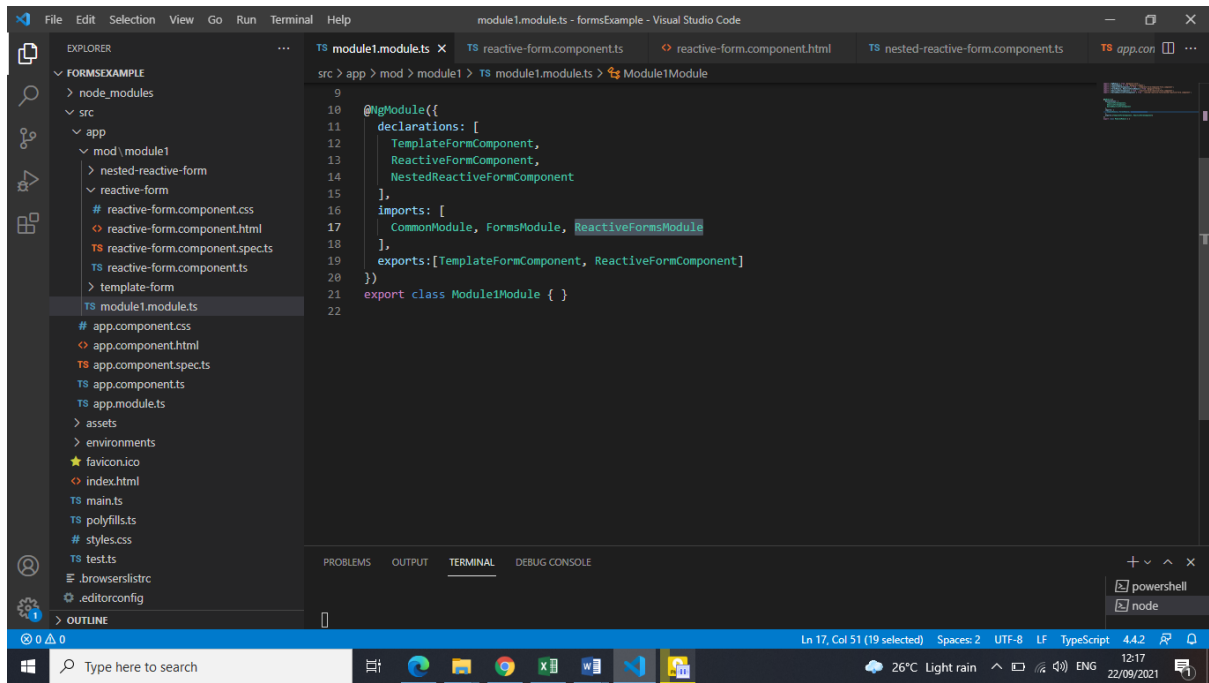
rely on directives in the template to create and manipulate the underlying object model. They are useful for adding a simple form to an app, such as an email list signup form. They're straightforward to add to an app, but they don't scale as well as reactive forms. If you have very basic form requirements and logic that can be managed solely in the template, template-driven forms could be a good fit.





2) Reactive Form:

provide direct, explicit access to the underlying forms object model. Compared to template-driven forms, they are more robust: they're more scalable, reusable, and testable. If forms are a key part of your application, or you're already using reactive patterns for building your application, use reactive forms.



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a project structure for 'FORMSEXAMPLE'. The main editor window shows the file 'reactive-form.component.html' with the following content:

```

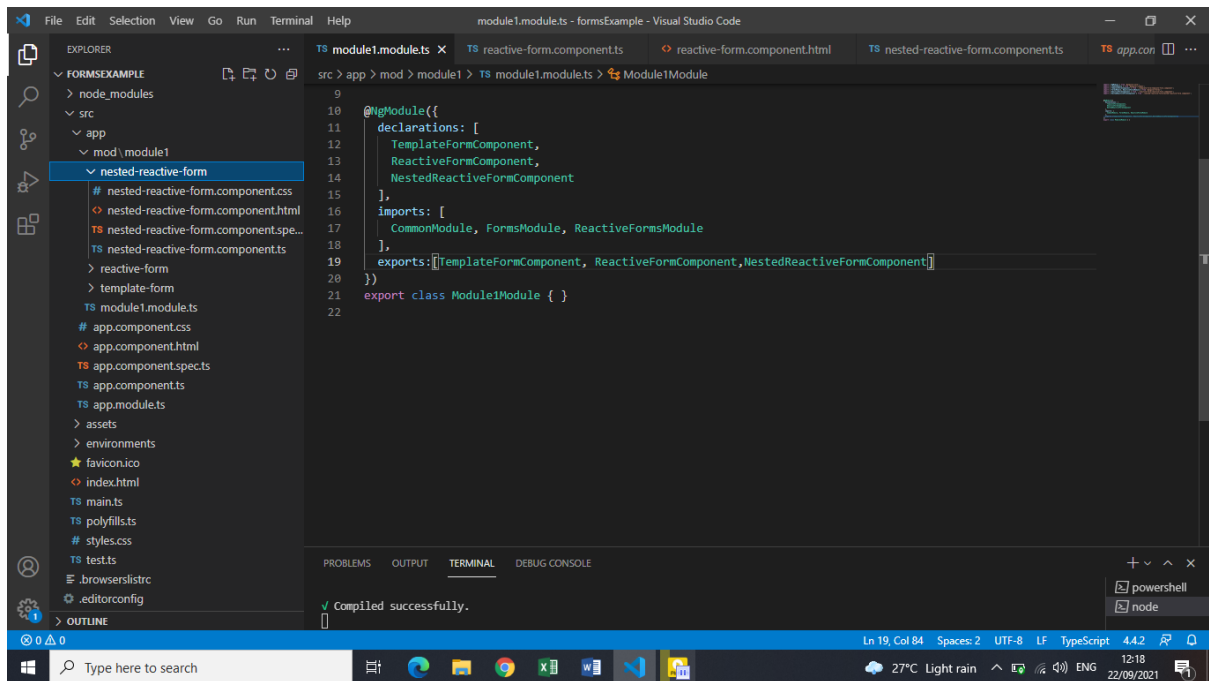
1 <p>reactive-form works!</p>
2
3 <form [formGroup]="loginForm" (ngSubmit)="callback()">
4   USERNAME: <input type="text" formControlName="username"><br>
5   PASSWORD: <input type="text" formControlName="password"><br>
6   <input type="submit" value="SUBMIT">
7 </form>
8

```

The bottom status bar indicates 'Compiled successfully.' and the terminal shows 'node'.

	REACTIVE	TEMPLATE-DRIVEN
Setup of form model	Explicit, created in component class	Implicit, created by directives
Data model	Structured and immutable	Unstructured and mutable
Data flow	Synchronous	Asynchronous
Form validation	Functions	Directives

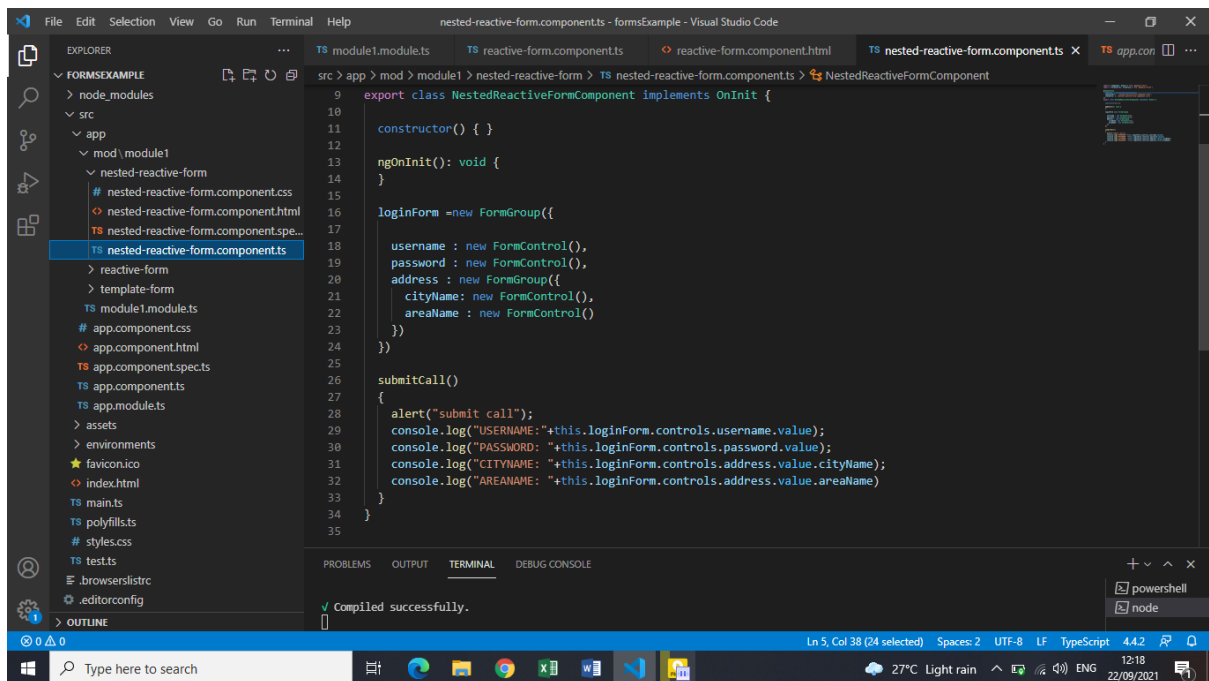
Nested Reactive Form:



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The project structure includes a 'FORMEXAMPLE' folder with subfolders 'node_modules', 'src', and 'app'. The 'src' folder contains 'mod' and 'app' subfolders. The 'mod' folder contains 'module1' and 'nested-reactive-form' subfolders. The 'nested-reactive-form' folder contains 'nested-reactive-form.component.css', 'nested-reactive-form.component.html', 'nested-reactive-form.component.spec.ts', 'nested-reactive-form.component.ts', 'reactive-form', 'template-form', 'module1.module.ts', 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', 'app.module.ts', 'assets', 'environments', 'favicon.ico', 'index.html', 'main.ts', 'polyfills.ts', 'styles.css', 'tests', 'browserlistrc', and 'editorconfig'. The 'module1.module.ts' file is open in the editor, showing the following code:

```
9
10 @NgModule({
11   declarations: [
12     TemplateFormComponent,
13     ReactiveFormComponent,
14     NestedReactiveFormComponent
15   ],
16   imports: [
17     CommonModule, FormsModule, ReactiveFormsModule
18   ],
19   exports: [TemplateFormComponent, ReactiveFormComponent, NestedReactiveFormComponent]
20 })
21 export class Module1Module { }
22
```

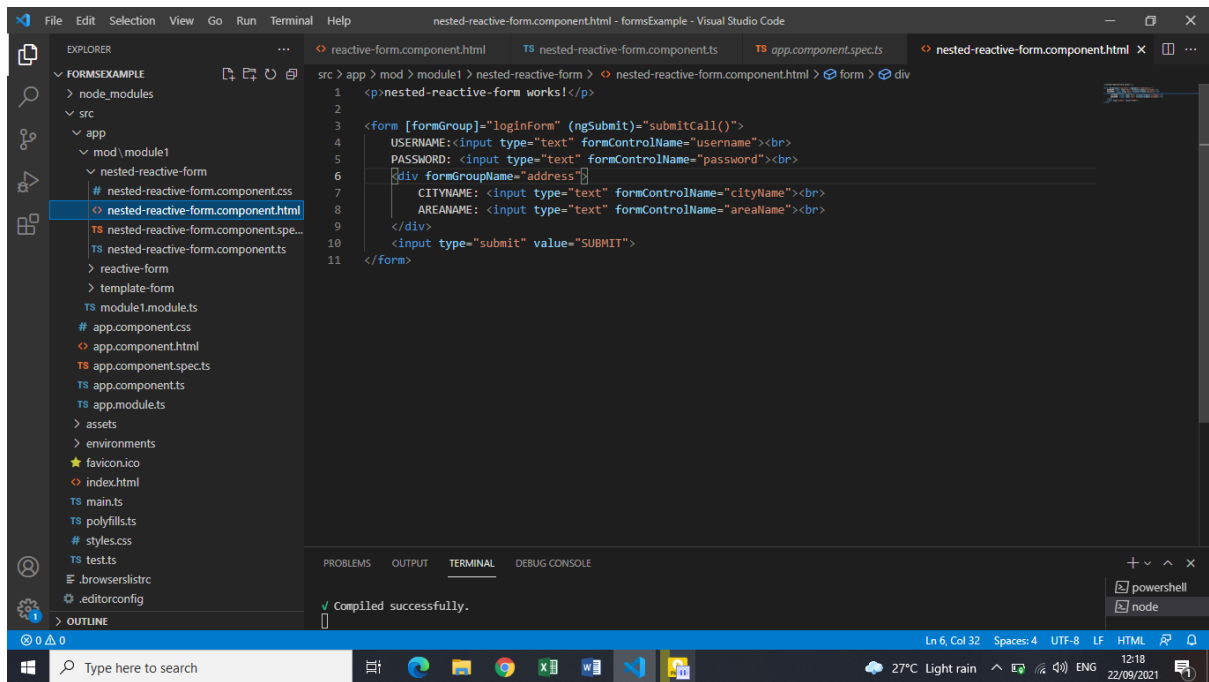
The status bar at the bottom indicates 'Ln 19, Col 84, Spaces: 2, UTF-8, LF, TypeScript, 4.4.2'. The terminal at the bottom shows 'Compiled successfully.'



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The project structure is the same as the previous screenshot. The 'nested-reactive-form.component.ts' file is open in the editor, showing the following code:

```
9 export class NestedReactiveFormComponent implements OnInit {
10
11   constructor() { }
12
13   ngOnInit(): void {
14
15   }
16
17   loginForm = new FormGroup({
18
19     username : new FormControl(),
20     password : new FormControl(),
21     address : new FormGroup({
22       cityName : new FormControl(),
23       areaName : new FormControl()
24     })
25   })
26
27   submitCall()
28   {
29     alert("submit call");
30     console.log("USERNAME: "+this.loginForm.controls.username.value);
31     console.log("PASSWORD: "+this.loginForm.controls.password.value);
32     console.log("CITYNAME: "+this.loginForm.controls.address.value.cityName);
33     console.log("AREANAME: "+this.loginForm.controls.address.value.areaName)
34   }
35 }
```

The status bar at the bottom indicates 'Ln 5, Col 38 (24 selected), Spaces: 2, UTF-8, LF, TypeScript, 4.4.2'. The terminal at the bottom shows 'Compiled successfully.'



```
1 <p>nested-reactive-form works!</p>
2
3
4 <form [formGroup]="loginForm" (ngSubmit)="submitCall()">
5   USERNAME:<input type="text" formControlName="username"><br>
6   PASSWORD:<input type="text" formControlName="password"><br>
7   <div formGroupName="address">
8     CITYNAME:<input type="text" formControlName="cityName"><br>
9     AREANAME:<input type="text" formControlName="areaName"><br>
10  </div>
11  <input type="submit" value="SUBMIT">
12 </form>
```

Compiled successfully.

Note: For more information use: <https://angular.io/guide/forms-overview>