

<b>Name of Student: Pushkar Sane</b>		
<b>Roll Number: 45</b>		<b>Practical Number: 10</b>
<b>Title of Lab Assignment: Demonstrate features of Angular forms with a Program.</b>		
<b>DOP: 23-09-2023</b>		<b>DOS: 24-09-2023</b>
<b>CO Mapped:</b> <b>CO5</b>	<b>PO Mapped:</b> <b>PO3, PO5, PSO1,</b> <b>PSO2</b>	<b>Signature:</b>

**Practical No. 10**

**Aim:** Demonstrate features of Angular forms with a Program.

**Description:**

For reference: YouTube playlists for explanation of the code

**Hindi:** [https://www.youtube.com/playlist?list=PL8p2l9GkIV45--5t7\\_N4lveUI6Y31vQ6C](https://www.youtube.com/playlist?list=PL8p2l9GkIV45--5t7_N4lveUI6Y31vQ6C)

(Watch videos #35 to #37)

OR

**English:** <https://www.youtube.com/playlist?list=PL8p2l9GkIV47eNpoo4Fr6fkags72a8F0v>

(Watch videos #34 to #37)

Angular is a popular web application framework that provides a variety of features for building dynamic and interactive web applications. Angular forms are a critical part of this framework, allowing developers to capture and handle user input. Here are some of the key features of Angular forms:

**1. Template-driven forms:**

Angular offers a template-driven approach for creating forms, which is suitable for simpler forms with less complex logic.

Forms are defined within the template HTML and are bound to the component using ngModel directives.

**2. Reactive forms (Model-driven forms):**

Reactive forms are more flexible and suitable for complex forms with extensive validation and custom logic.

They are defined programmatically in the component class, offering full control over form behavior and validation.

**3. Two-way data binding:**

In template-driven forms, Angular provides two-way data binding using ngModel, allowing data to flow seamlessly between the template and component.

**4. FormControl and FormGroup:**

Reactive forms use FormControl and FormGroup classes to represent form controls and groups of controls.

FormControl represents an individual input field, while FormGroup is used to group related form controls together.

**5. Validation:**

Both template-driven and reactive forms support client-side validation with built-in validators like required, minLength, maxLength, pattern, and custom validators.

Validation messages can be displayed based on the form's state and user interactions.

**6. Asynchronous validation:**

Reactive forms allow you to perform asynchronous validation, such as checking data on a server or making HTTP requests before allowing form submission.

**7. Dynamic form controls:**

You can dynamically add, remove, or modify form controls at runtime based on user interactions or other criteria.

**8. FormArray:**

Reactive forms include FormArray, which allows you to work with dynamic lists of form controls, like repeating form fields or dynamic form sections.

**9. Error handling:**

Angular provides mechanisms for handling and displaying form errors and validation messages, making it easier for users to understand and correct input issues.

**10. Form submission:**

Angular forms provide mechanisms for handling form submission, whether it's sending data to a server via HTTP requests or performing actions on the client side.

**11. ngSubmit:**

Both template-driven and reactive forms support the `ngSubmit` directive, which allows you to define a function to be executed when the form is submitted.

**12. Template-driven form features:**

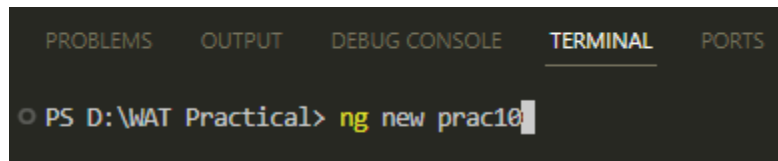
Features specific to template-driven forms include `ngModel`, `ngModelGroup`, `ngForm`, and `ngModelChange` for custom event handling.

**13. Reactive form features:**

Features specific to reactive forms include `FormBuilder` for simplifying form control creation, custom validators, value changes observables, and more control over the form's behavior.

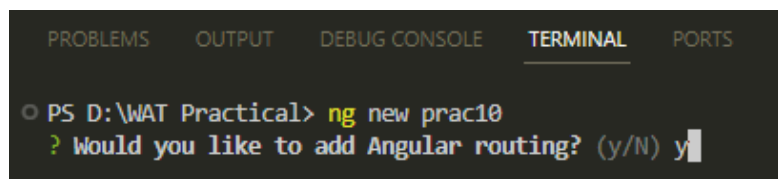
**1. Demonstrate features of Angular forms with a program****Steps to execute the practical:**

1. Create a folder "practical 10"
2. Open the terminal (please use cmd, avoid using powershell) and navigate the "practical 10" folder by using the `cd` command
3. Create a new angular project by running the following command  
`ng new prac10`



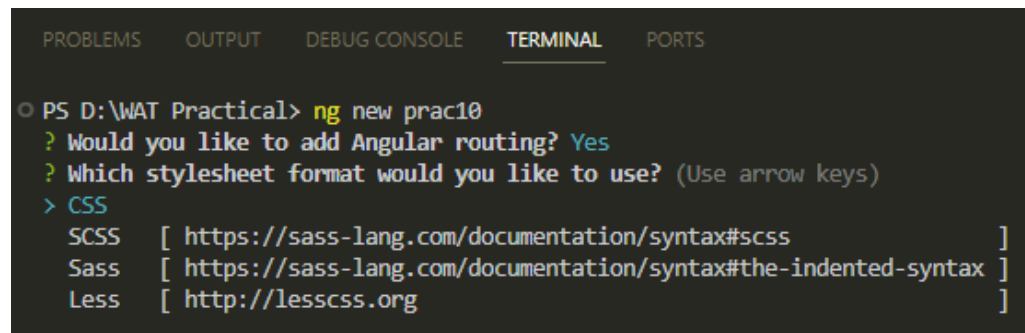
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\WAT Practical> ng new prac10
```

4. When prompted for Angular Routing, Enter "y"



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS D:\WAT Practical> ng new prac10
? Would you like to add Angular routing? (y/N) y
```

5. When prompted for CSS, use arrow keys to select “CSS” (it is the first option and is selected by default) and then press Enter



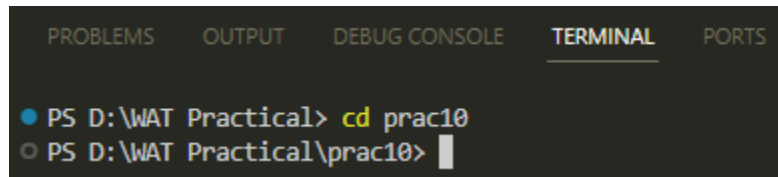
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\WAT Practical> ng new prac10
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS [ https://sass-lang.com/documentation/syntax#scss ]
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less [ http://lesscss.org ]
```

6. The new Angular project is created



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\WAT Practical> ng new prac10
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE prac10/angular.json (2700 bytes)
CREATE prac10/package.json (1037 bytes)
CREATE prac10/README.md (1060 bytes)
CREATE prac10/tsconfig.json (901 bytes)
CREATE prac10/.editorconfig (274 bytes)
CREATE prac10/.gitignore (548 bytes)
CREATE prac10/tsconfig.app.json (263 bytes)
CREATE prac10/tsconfig.spec.json (273 bytes)
CREATE prac10/.vscode/extensions.json (130 bytes)
CREATE prac10/.vscode/launch.json (470 bytes)
CREATE prac10/.vscode/tasks.json (938 bytes)
CREATE prac10/src/main.ts (214 bytes)
CREATE prac10/src/favicon.ico (948 bytes)
CREATE prac10/src/index.html (292 bytes)
CREATE prac10/src/styles.css (80 bytes)
CREATE prac10/src/app/app-routing.module.ts (245 bytes)
CREATE prac10/src/app/app.module.ts (393 bytes)
CREATE prac10/src/app/app.component.html (23115 bytes)
CREATE prac10/src/app/app.component.spec.ts (991 bytes)
CREATE prac10/src/app/app.component.ts (210 bytes)
CREATE prac10/src/app/app.component.css (0 bytes)
CREATE prac10/src/assets/.gitkeep (0 bytes)
✓ Packages installed successfully.
```

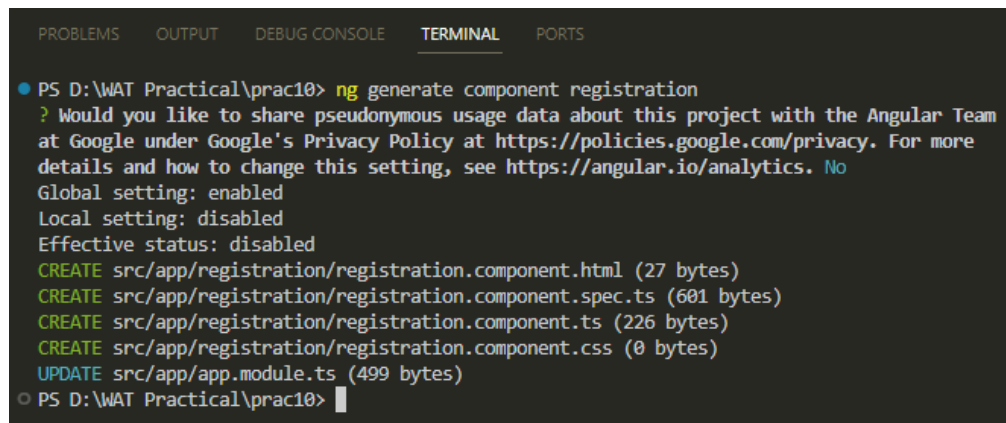
7. Navigate to the prac10 folder by using the cd command  
cd prac10



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\WAT Practical> cd prac10
PS D:\WAT Practical\prac10>
```

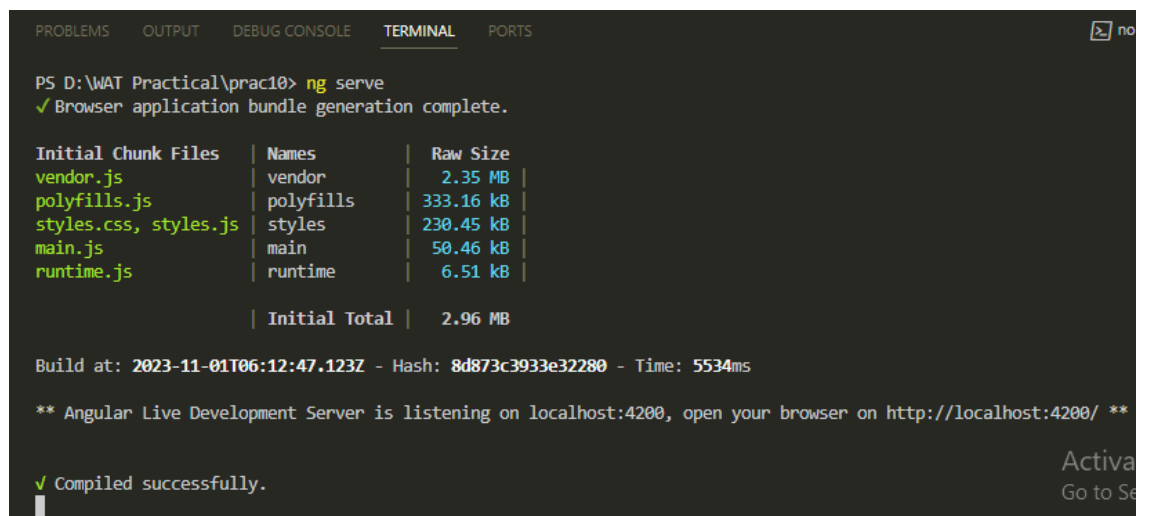
8. Create a new component “registration” by running the following command. This registration component will be our registration form  
ng generate component registration



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\WAT Practical\prac10> ng generate component registration
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: enabled
Local setting: disabled
Effective status: disabled
CREATE src/app/registration/registration.component.html (27 bytes)
CREATE src/app/registration/registration.component.spec.ts (601 bytes)
CREATE src/app/registration/registration.component.ts (226 bytes)
CREATE src/app/registration/registration.component.css (0 bytes)
UPDATE src/app/app.module.ts (499 bytes)
PS D:\WAT Practical\prac10>
```

9. Run the following command to serve the angular project on a server  
ng serve



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\WAT Practical\prac10> ng serve
✓ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size
vendor.js           | vendor | 2.35 MB
polyfills.js        | polyfills | 333.16 kB
styles.css, styles.js | styles | 230.45 kB
main.js             | main | 50.46 kB
runtime.js          | runtime | 6.51 kB
                    | Initial Total | 2.96 MB

Build at: 2023-11-01T06:12:47.123Z - Hash: 8d873c3933e32280 - Time: 5534ms

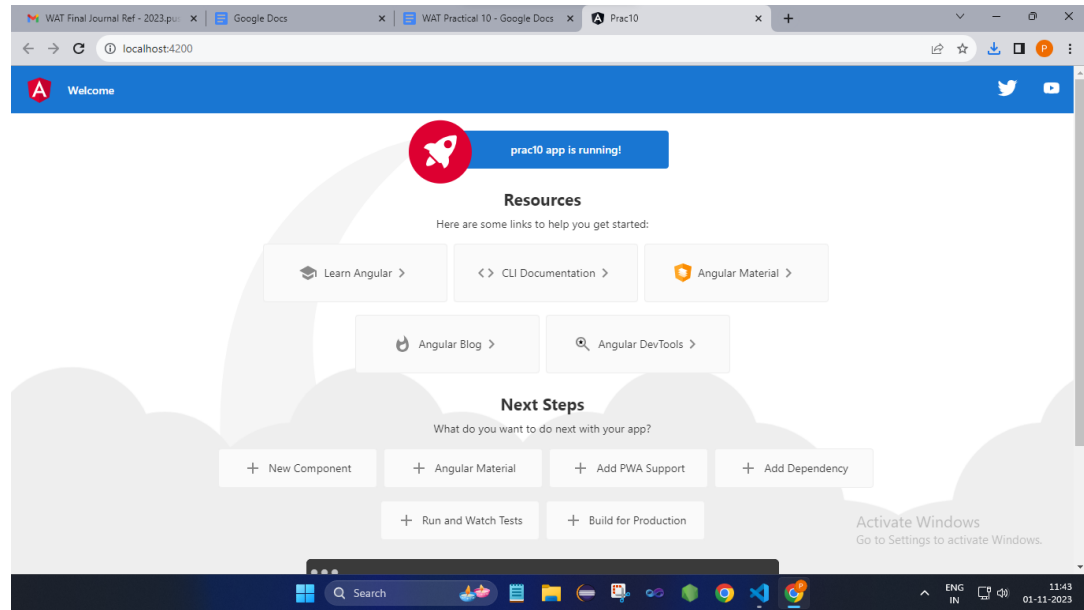
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

10. The server is running and will automatically re-compile the project when we make any changes

11. Open any browser and go to the following link to see the output (this is the default output whenever any new Angular project is made)

<http://localhost:4200/>



12. We will add the `ReactiveFormsModule` to the `app.module.ts` file in the `prac10/src/app/` folder

```
TS app.module.ts M X
prac10 > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { RegistrationComponent } from './registration/registration.component';
7
8  @NgModule({
9    declarations: [
10     AppComponent,
11     RegistrationComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule
16   ],
17   providers: [],
18   bootstrap: [AppComponent]
19 })
20 export class AppModule { }
21
```

```
TS app.module.ts M X
prac10 > src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { ReactiveFormsModule } from '@angular/forms'; // Import the ReactiveFormsModule
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6  import { RegistrationComponent } from './registration/registration.component';
7
8  @NgModule({
9    declarations: [
10     AppComponent,
11     RegistrationComponent
12   ],
13   imports: [
14     BrowserModule,
15     ReactiveFormsModule, // Add ReactiveFormsModule to the imports array
16     AppRoutingModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
22
```



After making the changes, the **app.module.ts** file should look like this:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ReactiveFormsModule } from '@angular/forms'; // Import the
ReactiveFormsModule
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { RegistrationComponent } from './registration/registration.component';
@NgModule({
  declarations: [
    AppComponent,
    RegistrationComponent
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule, // Add ReactiveFormsModule to the imports array
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

13. Add the “registration” component to the app.component.html file

Delete all the contents of **app.component.html** file (located in the prac11/src/app/ folder)

Add the following code to the file and save it

<app-registration></app-registration>

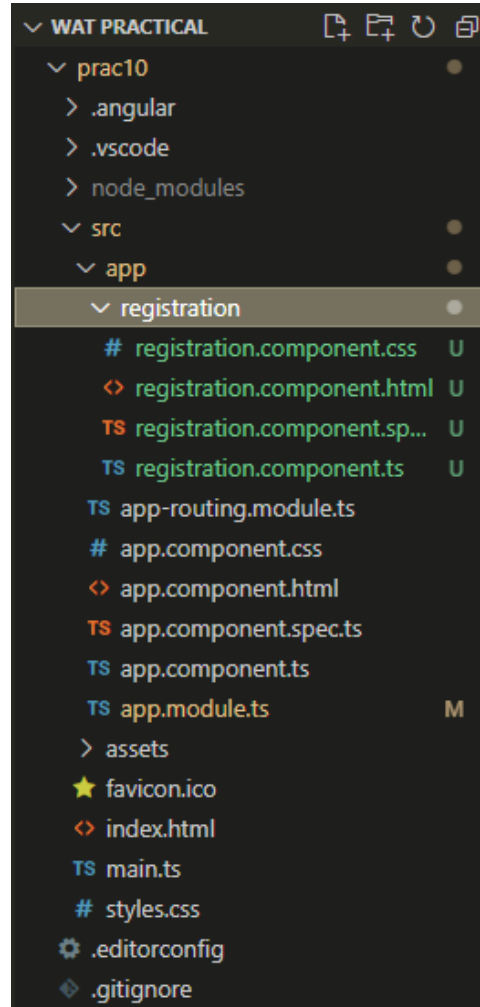
14. Now we'll make the registration form.

We will make changes to the following 3 files in the “registration” folder (prac10/src/app/registration/)

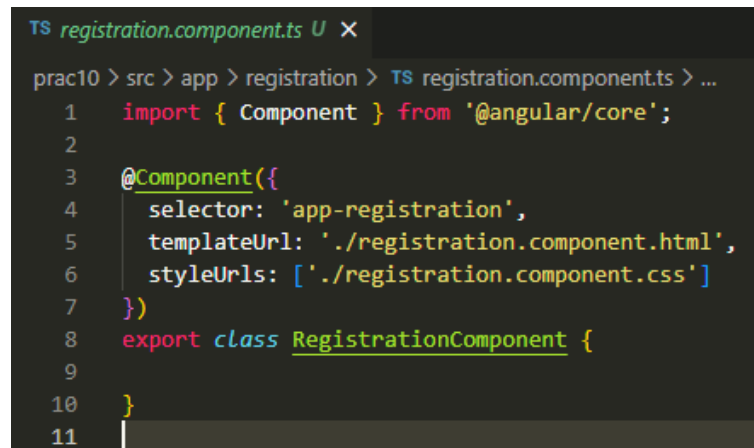
registration.component.ts

registration.component.html

registration.component.css



15. We'll modify the registration.component.ts file in the prac10/src/app/registration folder



```
TS registration.component.ts U X
prac10 > src > app > registration > TS registration.component.ts > RegistrationComponent
1  import { Component } from '@angular/core';
2  import { FormGroup, FormControl, Validators } from '@angular/forms'; // import required components
3  @Component({
4    selector: 'app-registration',
5    templateUrl: './registration.component.html',
6    styleUrls: ['./registration.component.css']
7  })
8  export class RegistrationComponent {
9    // added code
10   registrationForm = new FormGroup(
11     {
12       name: new FormControl('', Validators.required),
13       email: new FormControl('', [Validators.required, Validators.email])
14     }
15   );
16   onSubmit() {
17     console.log("Name: ", this.name?.value)
18     console.log("Email: ", this.email?.value)
19     // reset the form after the user has submitted the information
20     this.registrationForm.reset()
21   }
22   get name() {
23     return this.registrationForm.get('name');
24   }
25   get email() {
26     return this.registrationForm.get('email');
27   }
28 }
```

After making the changes, the **registration.component.ts** file should look like this:

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms'; // import
required components
@Component({
  selector: 'app-registration',
  templateUrl: './registration.component.html',
  styleUrls: ['./registration.component.css']
})
export class RegistrationComponent {
  // added code
  registrationForm = new FormGroup(
  {
    name: new FormControl("", Validators.required),
    email: new FormControl("", [Validators.required, Validators.email])
  }
  );
```

```
onSubmit() {  
  console.log("Name: ", this.name?.value)  
  console.log("Email: ", this.email?.value)  
  // reset the form after the user has submitted the information  
  this.registrationForm.reset()  
}  
get name() {  
  return this.registrationForm.get('name');  
}  
get email() {  
  return this.registrationForm.get('email');  
}  
}
```

16. Delete all the default generated code in registration.component.html file in the prac10/src/app/registration folder.

Write the following code in registration.component.html

```
<h1>Registration Form</h1>  
<form [formGroup]="registrationForm" (ngSubmit)="onSubmit()">  
  Name: <input type="text" formControlName="name">  
    <p class="error-message" *ngIf="name?.touched && name?.invalid">Name is  
invalid</p>  
  <br>  
  Email: <input type="email" formControlName="email">  
    <p class="error-message" *ngIf="email?.touched && email?.invalid">Email  
isinvalid</p>  
  <br>  
  <button [disabled]="registrationForm.invalid">Submit</button>  
</form>
```

17. Save the registration.component.html file.

18. Open the registration.component.css and add the following code in it:

```
.error-message {  
  color: red;  
}
```

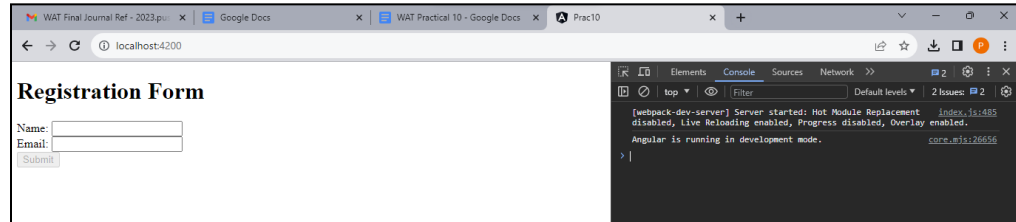
19. Save the registration.component.css file

20. Go to the browser to see the output at <http://localhost:4200/>

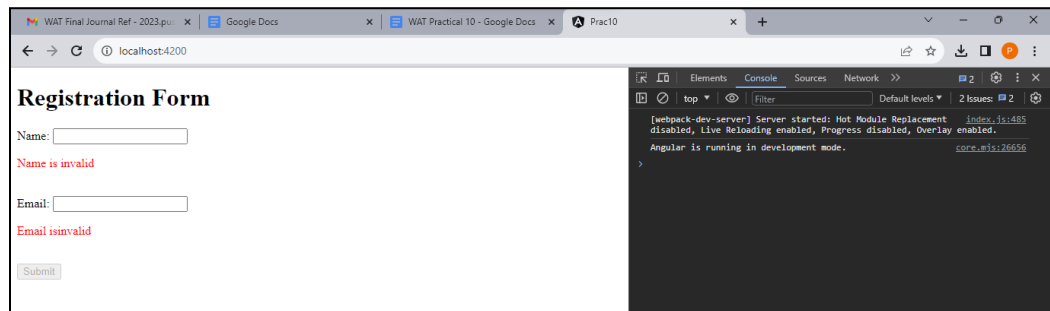
Press the following keyboard shortcut to see the console in the browser

Ctrl + Shift + J

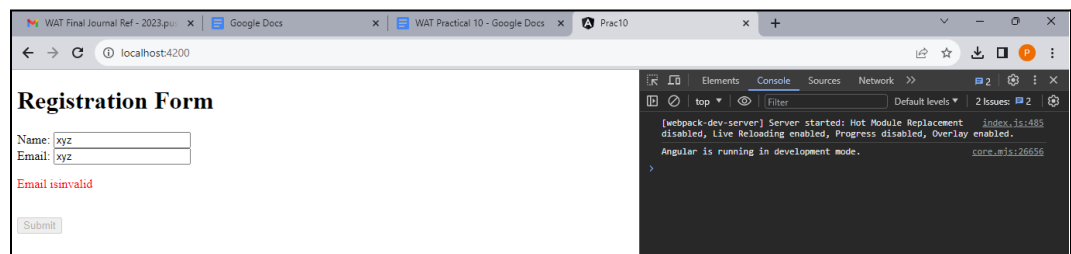
### Registration Form Output



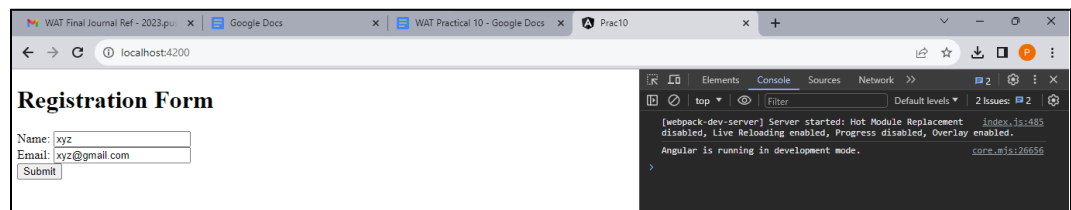
Click on the input boxes and then click outside the boxes to see the form validation



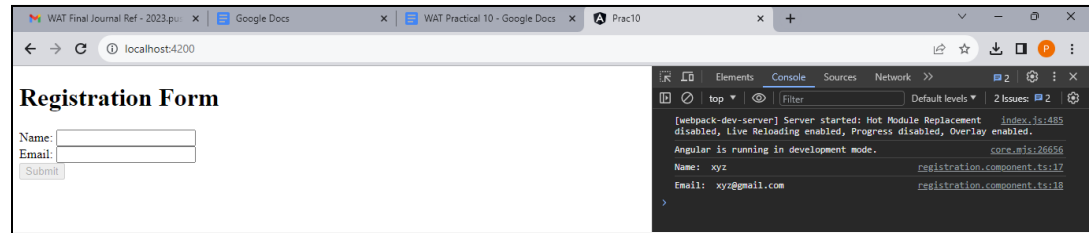
Type all valid input values to enable the Submit Button



Type all valid input values to enable the Submit Button

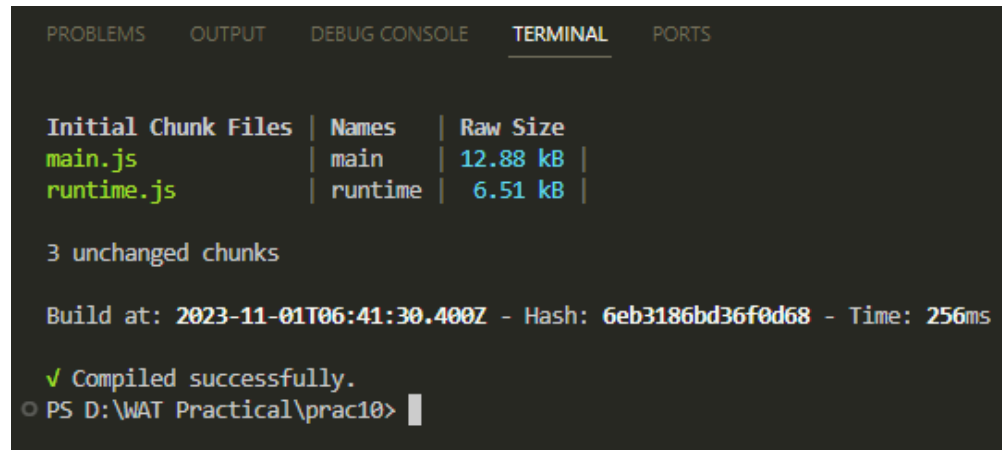


Submit the form to see the values in the console



21. Go to the terminal and stop the server by pressing Ctrl + C

When we see the prompt as "...\prac10>" then it means that the server has stopped



### Code:

#### **app.module.ts**

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { ReactiveFormsModule } from '@angular/forms'; // Import the
ReactiveFormsModule
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { RegistrationComponent } from './registration/registration.component';
@NgModule({
  declarations: [
    AppComponent,
    RegistrationComponent
```

```
],
imports: [
  BrowserModule,
  ReactiveFormsModule, // Add ReactiveFormsModule to the imports array
  AppRoutingModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

**app.component.html**

```
<app-registration></app-registration>
<router-outlet></router-outlet>
```

**registration.component.ts**

```
import { Component } from '@angular/core';
import { FormGroup, FormControl, Validators } from '@angular/forms'; // import required
components
@Component({
  selector: 'app-registration',
  templateUrl: './registration.component.html',
  styleUrls: ['./registration.component.css']
})
export class RegistrationComponent {
  // added code
  registrationForm = new FormGroup(
    {
      name: new FormControl("", Validators.required),
      email: new FormControl("", [Validators.required, Validators.email])
    }
  );
  onSubmit() {
    console.log("Name: ", this.name?.value)
```

```
console.log("Email: ", this.email?.value)
// reset the form after the user has submitted the information
this.registrationForm.reset()
} get name() {
  return this.registrationForm.get('name');
}
get email() {
  return this.registrationForm.get('email');
}
}
```

**registration.component.html**

```
<h1>Registration Form</h1>
<form [formGroup]="registrationForm" (ngSubmit)="onSubmit()">
  Name: <input type="text" formControlName="name">
    <p class="error-message" *ngIf="name?.touched && name?.invalid">Name is
invalid</p>
    <br>
  Email: <input type="email" formControlName="email">
    <p class="error-message" *ngIf="email?.touched && email?.invalid">Email
isinvalid</p>
    <br>
  <button [disabled]="registrationForm.invalid">Submit</button>
</form>
```

**registration.component.css**

```
.error-message {
  color: red;
}
```



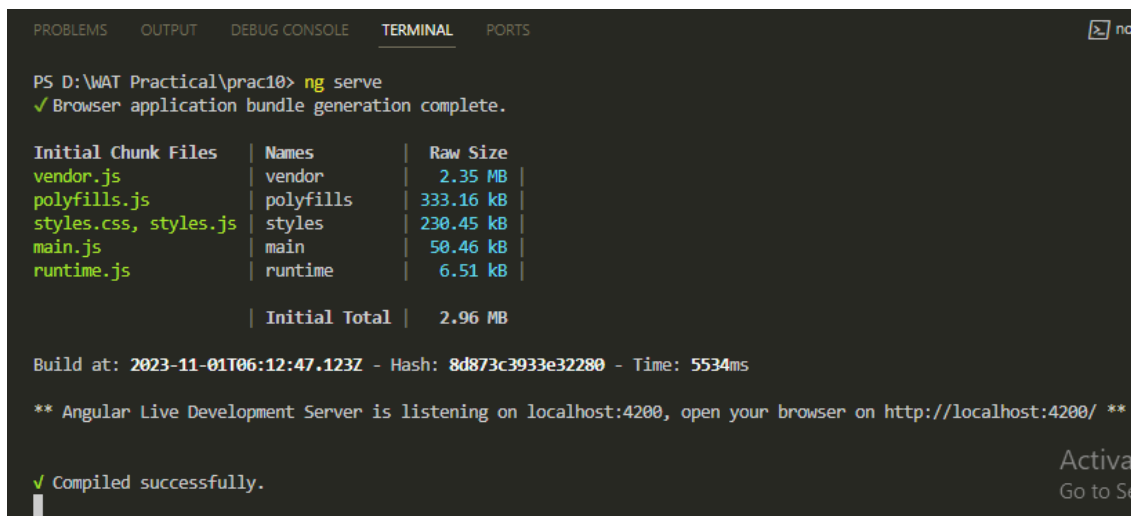
**Reference for explanation of the code**Hindi: [https://www.youtube.com/playlist?list=PL8p2I9GkIV45--5t7\\_N4IveUI6Y31vQ6C](https://www.youtube.com/playlist?list=PL8p2I9GkIV45--5t7_N4IveUI6Y31vQ6C)

(Watch videos #35 to #37)

OR

English: <https://www.youtube.com/playlist?list=PL8p2I9GkIV47eNpoo4Fr6fkags72a8F0v>

(Watch videos #34 to #37)

**Output:****Console**

```
PS D:\WAT Practical\prac10> ng serve
✓ Browser application bundle generation complete.

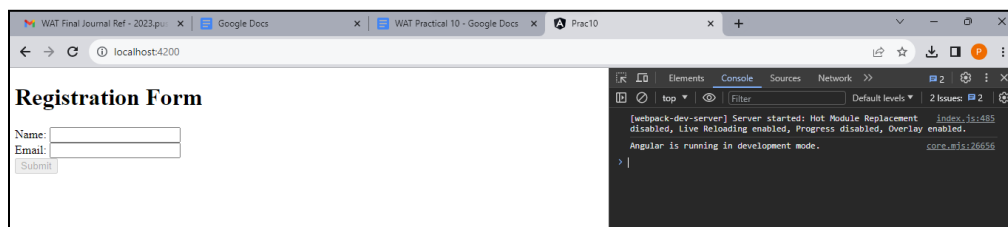
Initial Chunk Files | Names      | Raw Size
vendor.js           | vendor     | 2.35 MB
polyfills.js        | polyfills  | 333.16 kB
styles.css, styles.js | styles    | 230.45 kB
main.js             | main       | 50.46 kB
runtime.js          | runtime    | 6.51 kB

| Initial Total | 2.96 MB

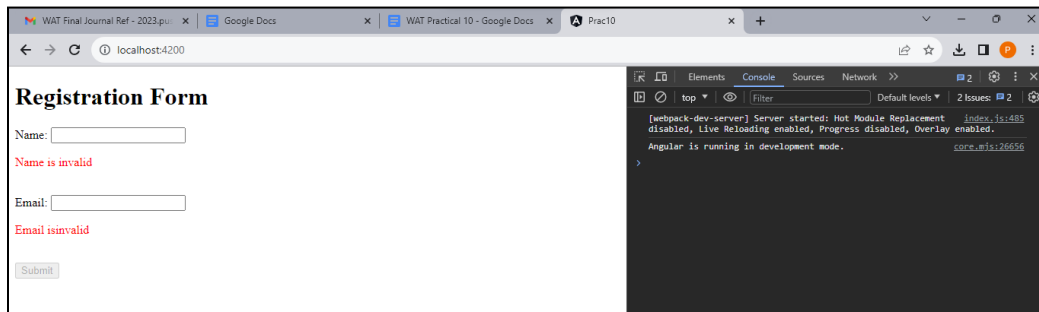
Build at: 2023-11-01T06:12:47.123Z - Hash: 8d873c3933e32280 - Time: 5534ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

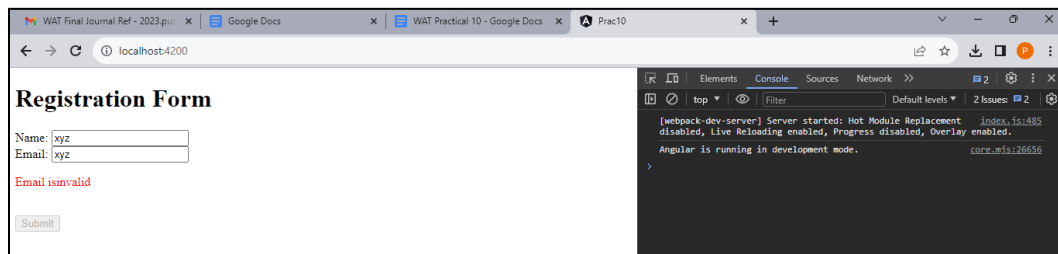
✓ Compiled successfully.
```

**Browser:****Press Ctrl + Shift + J to see the console****Registration Form Output**

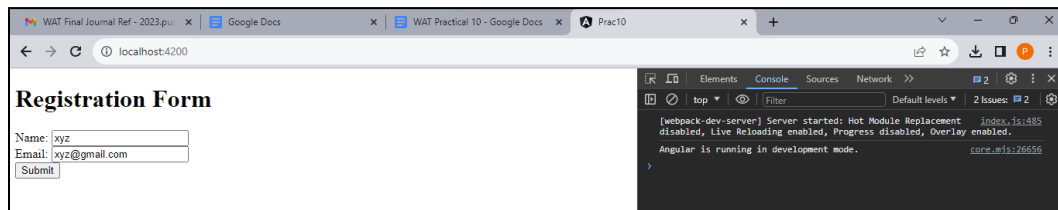
Click on the input boxes and then click outside the boxes to see the form validation



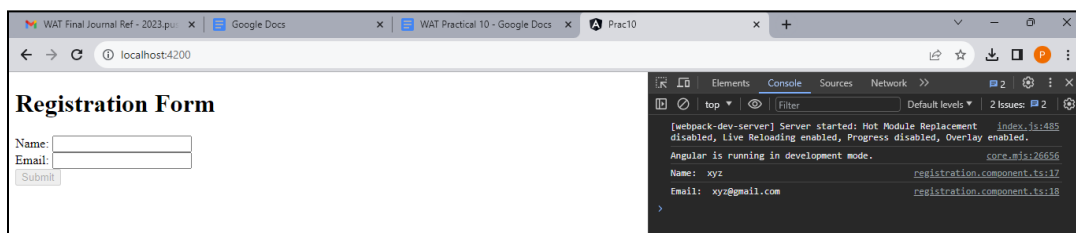
Type an invalid email to see the email's validation



Type all valid input values to enable the Submit Button



Submit the form to see the values in the console



**Conclusion:** We learnt about forms in Angular js.