SSGMCE	SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGG.	LABORATORY MANUAL	
	PRACTICAL EXPERIMENT INSTRUCTION SHEET		
	EXPERIMENT TITLE: DESIGNING AND WORKING WITH REACTIVE FORMS IN ANGULAR20		
EXPERIMENT NO.: **SSGMCE-WI-**IT-5IT09-08	ISSUE NO.: 00	ISSUE DATE: 30-07-2025	
REV. DATE:	REV. NO.:	DEPTT.: INFORMATION TECHNOLOGY	
LABORATORY: Computer Laboratory- 01	SEMESTER: V	PAGE :1 OF 4	

1.0) AIM: Designing and working with reactive Forms

2.0) SCOPE:

Designing and working with Model Driven Forms involves creating dynamic forms using TypeScript classes and decorators, managing form data binding, implementing validation, handling user interactions, and submitting form data. This approach is particularly useful when you need more control over the form's behavior and structure, and when you want to create complex forms with dynamic features.

3.0) FACILITIES/ APPARATUS:

i)Hardware: I3 Processor, 8GB RAM, HD Monitor, Windows 10, Internet connectivity

ii) Software: Updated Web browser, Node js, Angular CLI, VS Code.

4.0) THEORY:

Model-Driven Forms (also known as Reactive Forms) in Angular provide a more explicit and flexible way to manage form state and validation compared to Template-Driven Forms. They are especially useful when you need to handle complex form scenarios and integrate custom validation logic. Here's how to design and work with Model-Driven Forms in Angular, along with an example:

1. **Setting Up Angular Project**:

   Create a new Angular project: **ng new Rectiveforms**

   Navigate to the project directory: **cd Reactiveforms**

2. **Creating the Form Component**: Generate a new component for your form: **ng generate component reactiveform_demo**

3. **Designing the Form**: Open **reactiveform_demo.ts** in the **src/app** folder, and set up the Model-Driven in the component. **import { CommonModule } from '@angular/common';**

```
import { Component } from '@angular/core';
import { FormControl, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';
@Component({
  selector: 'app-reactiveform',
  imports: [CommonModule,ReactiveFormsModule],
  templateUrl: './reactiveform.html',
  styleUrl: './reactiveform.css'
})
export class Reactiveform {
// Step 2: Define the form model
  userForm = new FormGroup({
    firstName: new FormControl('', [Validators.required, Validators.minLength(3)]),
    email: new FormControl('', [Validators.required, Validators.email]),
    age: new FormControl('', [Validators.required, Validators.min(18)]),
    password: new FormControl('', [Validators.required, Validators.minLength(6)])
  });

  // Step 3: Handle form submission
  onSubmit() {
    if (this.userForm.valid) {
      console.log(this.userForm.value);
      alert('Form submitted successfully!');
    } else {
      alert('Please fill all required fields correctly.');
    }
  }
}
```

4. **Creating the Form Template**: Open **reactiveform_demo.html** in the same folder, and design the form using Angular's built-in form directive

```
<div class="container mt-5">

  <h2 class="mb-3">Reactive Form Example</h2>

  <form [formGroup]="userForm" (ngSubmit)="onSubmit()">

    <!-- First Name -->

    <div class="mb-3">

      <label class="form-label">First Name</label>

      <input type="text" class="form-control" formControlName="firstName">

      <div class="text-danger" *ngIf="userForm.controls['firstName'].invalid &&
userForm.controls['firstName'].touched">

        First Name is required (min 3 characters).

      </div>

    </div>

    <!-- Email -->

    <div class="mb-3">

      <label class="form-label">Email</label>

      <input type="email" class="form-control" formControlName="email">

      <div class="text-danger" *ngIf="userForm.controls['email'].invalid &&
userForm.controls['email'].touched">

        Please enter a valid email.

      </div>

    </div>

    <!-- Age -->

    <div class="mb-3">

      <label class="form-label">Age</label>

      <input type="number" class="form-control" formControlName="age">
```

```
<div class="text-danger" *ngIf="userForm.controls['age'].invalid &&
userForm.controls['age'].touched">

    Age must be 18 or above.

  </div>

</div>


<!-- Password -->

<div class="mb-3">

  <label class="form-label">Password</label>

  <input type="password" class="form-control" formControlName="password">

  <div class="text-danger" *ngIf="userForm.controls['password'].invalid &&
userForm.controls['password'].touched">

    Password must be at least 6 characters.

  </div>

</div>

<button class="btn btn-primary" type="submit"
[disabled]="userForm.invalid">Submit</button>

</form>

<!-- Debugging Form Values -->

<pre class="mt-4">{{ userForm.value | json }}</pre>

</div>
```

5. **Running the Application**: Run the development server using **ng serve**, and navigate to **http://localhost:4200** in your browser to see and interact with the form.

5.0) Conclusion:

We demonstrate the basics of working with Model-Driven Forms in Angular. we can extend this example by adding more form controls, custom validation logic, and handling form submission actions according to your application's requirements.