Below is the Angular **Login** and **Registration** UI with form validation, JWT storage in `localStorage`, and navigation after login.

---

1) Auth Service

``

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({ providedIn: 'root' })
export class AuthService {
  private API = '/api/auth';
  constructor(private http: HttpClient) {}

  login(data: { email: string; password: string }): Observable<{ token:
string }> {
    return this.http.post<{ token: string }>(`${this.API}/login`, data);
  }

  register(data: { name: string; email: string; password: string }):
Observable<any> {
    return this.http.post(`${this.API}/register`, data);
  }
}
```

---

2) Login Component

``

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';
import { TokenService } from '../../core/auth/token.service';

@Component({ selector: 'app-login', templateUrl: './login.component.html' })
export class LoginComponent {
  email = '';
  password = '';
  error = '';
  loading = false;

  constructor(private auth: AuthService, private token: TokenService,
private router: Router) {}
```

```
  submit() {
    if (!this.email || !this.password) {
      this.error = 'All fields are required';
      return;
    }
    this.loading = true;
    this.auth.login({ email: this.email, password:
this.password }).subscribe({
      next: res => {
        this.token.setToken(res.token);
        this.router.navigate(['/dashboard']);
      },
      error: err => {
        this.error = err.error?.message || 'Login failed';
        this.loading = false;
      }
    });
  }
}
```

``

```
<section class="wrap auth-form">
  <h2>Login</h2>
  <div *ngIf="error" class="error">{{ error }}</div>

  <label>Email <input type="email" [(ngModel)]="email" required /></label>
  <label>Password <input type="password" [(ngModel)]="password" required /
></label>

  <button (click)="submit()" [disabled]="loading">{{ loading ? 'Logging
in…' : 'Login' }}</button>
  <p>Don't have an account? <a routerLink="/register">Register</a></p>
</section>
```

---

3) Registration Component

``

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth.service';

@Component({ selector: 'app-register', templateUrl: './
register.component.html' })
export class RegisterComponent {
  name = '';
  email = '';
  password = '';
  confirmPassword = '';
  error = '';
  loading = false;

  constructor(private auth: AuthService, private router: Router) {}

  submit() {
    if (!this.name || !this.email || !this.password || !
this.confirmPassword) {
      this.error = 'All fields are required';
      return;
    }
    if (this.password !== this.confirmPassword) {
      this.error = 'Passwords do not match';
      return;
    }
    this.loading = true;
    this.auth.register({ name: this.name, email: this.email, password:
this.password }).subscribe({
      next: _ => this.router.navigate(['/login']),
      error: err => {
        this.error = err.error?.message || 'Registration failed';
        this.loading = false;
      }
    });
  }
}
```

``

```
<section class="wrap auth-form">
  <h2>Register</h2>
  <div *ngIf="error" class="error">{{ error }}</div>

  <label>Name <input [(ngModel)]="name" required /></label>
  <label>Email <input type="email" [(ngModel)]="email" required /></label>
  <label>Password <input type="password" [(ngModel)]="password" required /
></label>
  <label>Confirm Password <input type="password"
[(ngModel)]="confirmPassword" required /></label>

  <button (click)="submit()" [disabled]="loading">{{ loading ?
'Registering…' : 'Register' }}</button>
  <p>Already have an account? <a routerLink="/login">Login</a></p>
</section>
```

4) Styling

``

```
.auth-form { max-width: 400px; margin: auto; padding: 1rem; border: 1px
solid #ddd; border-radius: 8px; }
.auth-form label { display: block; margin-bottom: 0.75rem; }
.auth-form input { width: 100%; padding: 0.5rem; margin-top: 0.25rem; }
.error { color: red; margin-bottom: 1rem; }
button { padding: 0.5rem 1rem; }
```

---

5) Routing

Add these routes to ``:

```
{ path: 'login', component: LoginComponent },
{ path: 'register', component: RegisterComponent }
```

This gives you fully validated login and registration pages that store the JWT token in `localStorage` and redirect users after login.