SSGMCE/FRM/32-B

SSGMCE	SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGG.		**LABORATORY MANUAL**
	PRACTICAL EXPERIMENT INSTRUCTION SHEET		
	EXPERIMENT TITLE: CREATE, USE AND NAVIGATE COMPONENTS.		
EXPERIMENT NO.: **SSGMCE-WI-**IT-5IT09-02		ISSUE NO.: 00	ISSUE DATE: 01-07-2024
REV. DATE: 01/07/2025	REV. NO.:	DEPTT.: INFORMATION TECHNOLOGY	
LABORATORY: Computer Laboratory- 01		SEMESTER: V	PAGE :1 OF 3

1.0) AIM: To learn how to create, use, and navigate between components in an Angular 20 application.

2.0) SCOPE: This experiment helps students understand the component-based architecture of Angular. It enables the creation of reusable components, setting up routing, and navigating between them using Angular Router. It forms the foundation for building modular and maintainable applications.

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:

In Angular, components are the basic building blocks of the UI. Each component consists of:
- A TypeScript class (contains logic),
- An HTML template (defines view),
- And a CSS/SCSS file (styles the view).
Component Lifecycle: Each component in Angular goes through a lifecycle with hooks like ngOnInit(), ngOnDestroy(), etc.
Navigation: Angular provides a built-in RouterModule that enables navigation between components using routes (URLs).
Syntax to create a component:

```
ng generate component componentName
```

To enable routing:
Import RouterModule in app.config.ts
Define paths using provideRouter() or in routes array.

Procedure

1. Step 1: Create Angular Project

```
ng new ComponentExperiment
cd ComponentExperiment
```

PREPARED BY:	APPROVED BY:(H.O.D.)

2. Step 2: Create Components

```
ng generate component home
ng generate component about
ng generate component contact
```

3. Step 3: Define Routes

app.ts file

```
import { provideRouter, Routes } from '@angular/router';
import { HomeComponent } from './app/home/home.component';
import { AboutComponent } from './app/about/about.component';
import { ContactComponent } from './app/contact/contact.component';

const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: 'contact', component: ContactComponent },
];

export const appConfig = {
  providers: [provideRouter(routes)],
};
```

4. Step 4: Create Navigation Bar

app.html

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">My App</a>
  <ul class="navbar-nav mr-auto">
    <li class="nav-item"><a class="nav-link" routerLink="/">Home</a></li>
    <li class="nav-item"><a class="nav-link" routerLink="/about">About</a></li>
    <li class="nav-item"><a class="nav-link" routerLink="/contact">Contact</a></li>
  </ul>
</nav>
<hr />
<router-outlet></router-outlet>
```

5. Step 5: Add Sample Content in Each Component

home.html:
<h2>Welcome to the Home Page!</h2>
<p>This is the home component.</p>

about.component.html:
<h2>About Us</h2>
<p>This page tells you about our Angular experiment.</p>

contact.html:
<h2>Contact Us</h2>
<p>Here is our contact information.</p>A single-page web application with a navbar.
Clicking on "Home", "About", or "Contact" links loads different components without reloading the page.Demonstrates component creation and routing.

5.0) CONCLUSION: directory structure provides a clear and organized way to manage different aspects of our Angular project. As our application grows, we can create additional directories and files within the app directory to better organize our components, services, modules, and other elements.