Expt. No. 1 Date

Page No. 2

Create a Simple Angular application with 4 component and arrange these components as 2 rows & 2 columns with title on each components as Component 1, Component 2, Component 3 and Component 4.

app.component.html.

```
<div class = "grid-container">
<div class = "grid-row">
<app-component 1 class = "grid-item"></app-component 1>
<app-component 2 class = "grid-item"></app-component 2>
</div>
<div class = "grid-row">
<app-component 3 class = "grid-item"></app-component 3>
<app-component 4 class = "grid-item"></app-component 4>
</div>
</div>
```

app.component.ts

```
import { Component 1 Component } from './component 1';
import { Component 2 Component } from './Component 2';
import { Component 3 Component } from './Component 3';
import { Component 4 Component } from './Component 4';
@Component ({
    )}
export class AppComponent {
}
```

Expt. No. Date

Page No. 4

app.component.css

```
.grid-container {
    display: flex;
    flex-direction: column;
    align-items: center;
}
.grid-row {
    display: flex;
    flex-direction: row;
}
.grid-item {
    margin: 10px;
    padding: 20px;
    border: 1px solid #ccc;
}
```

component1.component.html

```
<h2> Component 1 </h2>
```

component2.component.html

```
<h2> Component 2 </h2>
```

component3.component.html

```
<h2> Component 3 </h2>
```

component4.component.html

```
<h2> Component 4 </h2>
```

Expt. No. 2 Date

Page No. 6

Create a parent and a child component & dimonstrate passing of value from parent to child & from the child to parent.

passing value from parent to child.

parent.Component.ts

```
import { Component } from '@angular/core';
@Component ({
    selector : 'app-parent';
    templateUrl : './parent.Component.html';
    styleUrls : ['./parent.Component.css']
})
export class parentComponent {
    parentValue = 'value from parent';
}
```

parent.Component.html

```
<h2> parent Component </h2>
<app-child. [childInput] = "parent value"> </app-child>
```

child.Component.ts

```
export class childcomponent {
    @Input() childInput : string;
}
```

Expt. No. .....

Date .....

Page No. 8

child.component.html

```
<p> Recieved value : {{childInput}} </p>
```

passing value from child to parent.

child.component.ts

```
export class ChildComponent {
  @Output() valueToParent = new EventEmitter<string>();
  SendValueToParent(){
    this.valueToParent.emit('value from child');
  }
}
```

parent.component.ts

```
export class ParentComponent {
  parentValue = 'value from parent';
  getValueFromChild(value:string){
    Console.log(value);
  }
}
```

Expt. No. 3 Date

Page No. 19

Create a text input in the parent component & pass Number data to the child component. In the child component, create a custom directive to change the font size based on the number passed from parent.

app.component.ts

```
export class AppComponent {
  size: number = 18;
  sendFontSize(value: string){
    this.size = parseInt(value);
  }
}
```

app.component.html

```
<input type="text" id="font-size" name="font-size" placeholder
="Enter the font size" #fontsize>
<button (click) = sendfontSize(fontsize.value)>Submit
</button>
<app-child-font [size]="size"></app-child-font>
```

child-font.component.ts

```
export class ChildFontComponent{
  @Input() size: number = 10;
}
```

Expt. No. .....

Date .....

Page No. 12

child - font.component.html

```
<div> The current font size : {{ size }} </div>
<h2 [appFontSize] = "size"> This is a sentance whose
font size changes !! </h2>
```

font-size.directive.ts

```
@Directive ({
    selector : '[appFontSize]',
    standalone : true
})

export class FontSizeDirective implements OnChanges{
  constructor (private el : ElementRef) { }
@Input ('appFontSize') fontSize : number = 0;

ngOnchanges (changes : SimpleChanges ['fontSize'].
  currentValue;
  if (newFontSize) {
    this.el.nativeElement.style.fontSize = ' ${ newFont
    size } px ';
}
}
}
```

Expt. No. 4 Page No. 14 Date

Create a Custom pipe to display first two characters of string.

character-string.pipe.ts

```
export class characterstring pipe, implements
    PipeTransform {
    transform (value: string, ...args: unknown[]):
    unknown {
        return value.substring(0,2);
    }
}
```

app.component.ts

```
import { characterString pipe } from './character-string.pipe';
@Component({
    imports: [characterstring pipe],
})

export class AppComponent{
    Name : String = 'MITE';
    title = 'lab programs';
}
```

app.component.html

```
<h2> String before passing into pipe{{Name}} </h2>
<h2> String after passing into Pipe {{Name| characterString}}</
```

Expt. No. 5 Date

Page No. 16

Create a Simple angular form with a 4 text fields and perform a mandatory check using React Form validation.

app.component.html.

```
<app-my-form (formSubmitted)="handleFormSubmission($event)"></app-my-form>
```

app.component.ts.

```
export class Application {
    handleFormSubmission(formData: any) {
        console.log("Form data Submitted:", formData)
    }
}
```

myform.component.html.

```
<form [formGroup]="myForm" (ngSubmit)="SubmitForm()">
<div>
<label for="firstName">First Name:</label>
<input type="text" id="firstName" formControlName="firstName">
<div *ngIf="firstNameInvalid" class="error-msg">First Name is required</div></div>
<div>
<label for="lastName">Last Name:</label>
<input type="text" id="lastName" formControlName="lastName">
```

```
<div *ngIf = "lastName Invalid" class="error-msg">Last
Name is Required </div>
</div>
<div>
<label for="email">Email: </label>
<input type="email" id="email" formControlName="email">
<div *ngIf="email Invalid" class=" error-msg ">
Email is Required </div>
</div>
<div>
<label for ="phone"> phone : </label>
<input type ="text" id="phone" formControlName="phone">
<div *ngIf = "phone Invalid" class ="error-msg"> Phone is
Required </div> </div>
<button type = "Submit" [disabled] = "myForm. invalid">
Submit </button>
<div> Form Status : {{ myForm. Status }} </div>
</form>
```

my-form.Component.ts.

```
export class MyFormComponent{
@Output () formSubmitted = new EventEmitter <any>();
myForm: FormGroup;
Constructor (private formBuilder: Form Builder){
  this.myForm = this formBuilder.group ({
    firstName : ['', validators.required],
    lastName : ['', validators.required],
    email : ['', validators. required],
```

Expt. No. ..... Date .....
Page No. 20

```
    phone: ['', Validators.required]
  });
}
get firstNameInvalid() {
  return this.myForm.get('firstName')?.invalid;
}
get lastNameInvalid() {
  return this.myForm.get('lastName')?.invalid;
}
get emailInvalid() {
  return this.myForm.get('email')?.invalid;
}
get phoneInvalid() {
  return this.myForm.get('phone')?.invalid;
}

submitForm() {
  if (this.myForm.valid) {
    this.formSubmitted.emit(this.myForm.value);
  } else {
    console.log('Form has errors');
  }
}
}
```

Expt. No. 6 Date

Page No. 22

Creat an application with 3 pages as Home, About and Contact and dimonstrate routing from one page to another on click of button in each page.

app.Component.html

```
<h1> <a raterlink="/home"> Home </a> </h1>
<h1><a routerlink="/about"> About </a></h1>
<h1><a routerlink="/contact"> Contact </a> </h1>
<router-outlook></router-outlet>
```

app.routes.ts

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

export const routes: Routes =[
  { path: 'home', Component: HomeComponent },
  { path: 'about', component: About Component},
  { path: 'Contact', Component: Contact Component},
];
```

home.Component.html.

```
<p> Home workssss !!</p>
```

Expt. No. .....

Date .....

Page No. 24

Contact.Component.html

. Contact worksss!!

About.Component.html

 About worksss!!.

Expt. No. 7 Date

Page No. 26

Create an application demonstrate the working of observable and subscriber using RxJs library with simple array.

app.html

```
<app-number-list></app-number-list>
```

number-list.Component.html

```
<button (click)="SubscribeTonumbers()">
Subscribe to Numbers </button>
```

number-list.Component.ts

```
import { Component, OnInit } from '@angular/core';
import { Observable of } from 'rxjs';

export class NumberListComponent implements OnInit {
numbers $: observable <number [] > = of ([])
Constructor () { }
ng onInit (): void {
  this.numbers $ = of ([1,2,3,4,5]);
}
Subscribe To Numbers () {
  this.numbers $. Subscribe (number => {
    console.log (numbers);
  });
}}
```

Expt. No. 8 Date

Page No. 28

Create an application with a button & onclick of button dimonstrate fetching data from any service and display the details.

app.component.html

```
<app-user> </app-user>
```

user.component.html

```
<div *ngIf = "userData">
<from (submit)= "onSubmit($event)" style = "border:1px
Solid black; border_radius:5px;>
<label for = "fullName"> Name: </label> <br>
<input type = "text" id = "fullName" [value] = "userData.name
.title + ' ' + userData.name.first + ' ' + userData.name.
last " disabled style = "bottom:10px;' > <br>
<label for = "email"> Email : </label> <br>
<input type = "text" id = "email" [value] = userData.email"
disabled > <br>
<label for = "phone"> phone : </label> <br>
<input type = "text" id = "phone" [value] = userData.
phone" disabled > <br>
<label for = "user photo" [src] = "userData.pictures.
large "alt = "user photo "> <br>
<button type = "submit "> submit </button>
</form>
</div>
```

Date .....

Page No. 30

Expt. No. .....

user.component.ts.

```
import { CommonModule } from '@angular/common';
import { HttpClient, HttpClientModule } from 'angular/
common/http';
import { Component } from @angular/core';
interface User {
    name : {
        title : string;
        first : string;
        last : string;
    };
    picture : { large : string ; };
    email : string;
    phone : string; }
    @Component ({
        selector : 'app-user';
        standalone : true;
        imports : [HttpClientModule, CommonModule]
        templateUrl : './user.component.html',
        styleUrl : './user.component.css'
    })
export class UserComponent {
    apiUrl = 'https://randomusers.me/api/?nat.in;
    userData : User/null = null;
    Constructor (private http : Http client){
        this.fetchUser();
}
```

```
fetchUser(){
  this.http.get<any>(this.apiUrl).subscribe({
  next:(response)=>{
  this.userData = response.results[0];
  console.log(this.userData);
};
  error:(err)=>{
  console.log('error fetching user', err);
  }
})
}

onSubmit(event: Event){
  event.preventDefault();
  console.log('Form Submitted:', this.userData);
}
}
```