WP

R. Sai chand
18K81A05I0
CSE-C

Assignment No. II Unit No. II Date :

1) Explain reactive programming briefly? - considerable operators.

Reactive Programming:-

Reactive programming is a programming paradigm dealing with datastreams and the propagation of changes. Data streams may be dynamic or static.

Reactive programming enables the datastream to be emitted from source called observable and emitted data stream to be caught by other source called observer through a process called subscription.

Javascript doesnot have the built-in functions for Reactive Prog. Angular uses Rxjs library extensively to do below mentioned advanced concepts:

- Data transfer between components
- HTTP client
- Router
- Reactive forms.

Observable:

Observable are data sources and they may be static or dynamic. Rxjs provides lot of method to create observable from common Javascript objects.

```
const number$ = of(1,2,3,4,5,6,7,8,9,10);
```

Subscribing process:

Subscribing to an observable is quite easy. Every observable object will have method

subscription pi Subscribe for the subscription process.

Observer need to implement three call back function to subscribe.

- next - Receive and process the value emitted from the observable
- error - Error handling call back.
- complete - call back function called when all data from observable are emitted.

Operations :-

Rxjs library provide some of the operators to process the data stream.

filter : Enable the filter the data stream using callback function.

```
const filterFn = filter ((num : number) => num>2);
const filteredNumbers$ = filterFn(numbers$);
filteredNumbers$.subscribe (num : number) =>
{this.filteredNumbers.push(num);
this.val2 += num});
```

map : Enables to map the data stream using callback function and to change the data stream itself.

```
const mapFn = map ((num : number) => num*num);
const mappedNumbers$ = mappedFn(numbers$)
```

Pipe :- Enable two or more operators to be combined.

```
const filterFn = filter ((num : number) => num>2);
```

```
const filteredNumber$ = filterFn(number$);
filteredNumber$.subscribe((num : number) =>
{this.filteredNumbers.push(num);
this.value+=num});
```

2) Explain Http client programming and briefly discuss expense Rest API.

Http client programming is a must needed feature in every modern web applications. Now adays, lot of applications exposes their functionality through Rest API.

Angular provides a separate module, HttpClient module and a service.

Http client to do HTTP programming

Expense Rest API :—

The Prerequiste to do Http programming is the basic knowledge of Http protocol and REST API technique

Http programming involves two part, server and client.

Angular provides support to create client side application.

Express a popular web framework provides support to create serverside application

Open a command prompt and create a new folder, express-rest-api

cd/go/to/workspace
mkdir/express-rest-api
cd Expense-rest-api
Initialize a new node application using below command:
npm init.

3. Explain the process of configure routing create and accessing routes with examples.

Configure routing:

Angular CLI provides complete support to setup routing during the application creation process as well as during working an application. Let us create a new application with routes enabled using below command ng new routing app-routing.

Angular CLI generate a new module, App Module for routing purpose. The generate code as follows:

```
import {NgModule} from '@angular/core';
import {Routes, RouterModule} from '@angular/router';
const routes: Routes = [];
@NgModule({
    imports: [RouteModule.forRoot(routes)];
    exports = [RouterModule].
})
export class AppRoutingModule{}
```

Assignment No. II Unit No. Date :

Creating routes:

Creating a route is simple and easy. The basic information to create a route is given below:

- Target component to be called.
- The path to access the target component.

The code to create a simple route is mentioned below:

```
const routes = Route = [
    { path: 'about', 'Component: About Component }
];
```

Accessing routes:

Accessing the route is a two step process.

Include router-outlet tag in the root component template

```
<router-outlet></router-outlet>
```

Use routerlink and routerLine Active property in the required place.

```
<a routerlink = "/about" routerLink Active"= "active"> First component</a>
```

- routerlink set the route to be called using the path.
- routerlink Active set the css class to be used when the route is activated.

4) Explain how to figure configure forms, create forms.

Configure Forms :-

To enable template driven forms, Let first we need to import FormsModule in app.module.ts. It is given below :

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing-module';
import { AppComponent } from './app.component';
import { FormsModule } from '@angular/forms';
imports : [
    BrowserModule,
    AppRoutingModule,
    FormsModule
],
```

Once Forms module is imported, the application will be ready for programming.

Create simple form

Let us create a sample application (template-form-app) in Angular 8 to learn the template driven form

Open command prompt and create new Angular application using below

Command : cd/go/to/workspace

```
ng new template_form-app
cd template-form-app
```

Assignment No. Unit No. Date :

Configure FormsModule in App component as shown below:

```
import { FormsModule } from '@angular/forms';
@NgModule({
  declarations: [
    AppComponent,
    TestComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Create a test component using Angular CLI as mentioned below:

```
ng generate component test.
```