```
app. routes. ts.

const My_ROUTES : Routes = [
{ path : 'home' Component : HomeComponent}, //
{ path : 'recent' Component : RecentComponent,
  children : [
  { path : 'transaction/:id' Component TransactionComponent}
},
{ path : 'dispute/:id' redirectTo : 'home' }

path : 'payment' Component PaymentComponent }

path : 'thankyou' Component ThankyouComponent }
```

```
app. module. ts.

declarations : [
    AppComponent,
    HomeComponent,
    RecentComponent,
    TrancactionComponent,
    PaymentComponent,
    ThankyouComponent

]
Imports : [



providers : [ DataService ],

bootstrap : [ MY-component ]
```

app. component. ts.

```
@Component.
    selector: 'app-root'
    template: '
<a [routerLink]="['home']" > Home </a>.
<a [routerLink]="['recent']" >View Recent Transacti</a>
<a [routerLink]="['payment']">Make Payment </a>

    <router-outlet> </router-outlet> //.

    '

export class AppComponent {




}.
```

data.service.ts.

```
export class DataService {
Constructor(private http:HttpClient, private route:Router){

}.
getUserInformation ( ) {
this.http.get("https://bank.com/api/home").
Subscribe ( (data) => {

    });
}.
getPending Transaction () {.
 this.http.get("https:// bank com/api/ recent").
 pipe(
    filter((data)=> data.status ='pending')
 ).
  Subscribe ( (responseData) => { } );

}
getApproved Transaction () {.
 this.http get("http://bank.com/api/recent').
 pipe(
    filter((data)=> data.Status='approved')
  ).
  Subscribe ((responseData) => {


      });
}.
```

```
getTransactionDetail(id:number){.
this.http.get("https://bank.com/api/recent/
            transaction/" +id).
 Subscribe ( (data) => {  } );
}.
makePayment(amount :number,
            bankno :number,
            date : date ) {.

const Payment = : {
  amount : 'amount'
  accountNumber : 'bankno'
  paymentDate : 'date' }

this.http.post("http://bank:com/api", Payment
.Subscribe(() => {
   this.route.navigate(['thankyou']);
  }
);
}.
getAccountNumber( id :number){.
this.http.get("http://bank.com/api/validationaccount/
            + id).
Subscribe (() => {
       }
    );
}.
```

home.Component.ts.

```
@ component ( {
    selector : 'home',
    template :``
  <p> Welcome Back {{userInfo.name}} </p>
  <div >
     <p> your Current balance : $ {{userinfo.balance}}</p>
    <p> Total expense : $ {{userinfo. expenses}}</p>
   <p>Last Payment : ${{userInfo. lastpayment}} </p>
</div>.
< pie [data]= "userinfo.chart" ></pie >



export class HomeComponent {
 const userInfo ;
 constructor(private dataService : DataService){}


}
OnInit () {.
this.dataservice. getUserInformation(). subscribe(
 (data) => {this. userinfo = data })

}.
```

```
Recent.component.ts.

@component ( {
  Selector : 'recent'
  template : `
<h2> Pending Transaction </h2>
<table >
   <tr>
     <th> Date </td>
     <th> Description </td>
     <th> category </td>
     <th> Amount </td>.
   </tr>.
  <tr *ngFor"let pending of pendingTransaction">
    <td>{{pending. date}} </td>
    <td>{{pending. description}}</td>
    <td> {{pending. category}}</td>
    <td> {{ pending amount}} </td>
  <td>
  <button [routerLink]="['transaction', id.value]"
   Details </button> </td>
  <td>
 <button [routerLink]="['dispute', id.value]"
   Dispute This charge  </button> </td>

</table >.
<h2> Approved Transaction </h2>.
<table >
   <tr>
     <th> Date </th>
     <th> Description </th>
     <th> Category </th>
     <th> Amount </th>.
  </tr>.
  <tr *ngFor=' let approve of approvedTransaction">
    <td> {{approve. date }} </td>
    <td> {{approve. discription }} </td>.
    <td> {{approve. category }} </td>
    <td> {{ approve amount }} </td>.
```

```
export class RecentComponent {
const approvedTransaction [];
const pendingTransaction [];
Constructor (private dataService : DataService){


}

OnInit () {
 this.dataService.getPendingTransaction().
 Subscribe((data) => {
      this.pendingTransaction = data;
                 }
   );
 this.dataService.getApprovedTransaction().
 Subscribe ((data) => {
        this.approvedTransaction = data;
              }
      )
}.
```

```
<td>
  <button [routerLink]="['transaction', id.value]"
        Details    </button
<td>.

<td>.
  <button [routerLink]="['dispute', id.value]"
     Dispute This charge.  </button>.
</td>

</table >

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

transaction. Component. ts.

```
@ component ( {

      Selector : transaction ,
      template : '
    <h1> Payment Detail's </h2>
  <div>
    <P1>{{ transactiondetail. discription }} </p>

   <p> {{transaction detail. address }} </p>

  < map [data] = "transactiondetail. Location"}} >.

  < / map >.

   <p> Transaction amount : {{transactiondetail. amount}}</p>
  <p> Transaction Date : {{transactiondetail. date }}</p>
  <p> Catgory : {{transactiondetail. Catgory}} </p>
 <p> Purchase Method : {{ transactiondetail. method}} </p>
 <p> Card used : {{ transactiondetail. card }}</p>
</div> '
@ export class TransactionComponent {

   const transactiondetail;
   const id,
 Constructor( priate dataService . DataService ;
             private ( route : ActivatedRoute )

   }

 OnInit () {
 this. route. params, subscribe ( param => {
     this. id = Params {'id'}});

this. dataservice. getTransactionDetail ( id ).

subscribe ( (data) => {
       this. transaction detail = data

    }
 )
}.
```

payment. Component - ts.

```
@Component ( {
    Selector : 'payment'
    template :

<form [formGroup] = 'my form'  (ngSubmit) = "OnPayment()">

 <dir >.
  <label  for = "Amount" > Amount </label>
  <input  type = 'text'
          name = 'amount'
          formControlName = 'amount'
          placeholder = '$' >.
<label  for = "bankNo" > Bank Accont Number </label>
<input  type = 'text'
        name = 'bankNO'
        formControlName = 'bankNO' >.
 (*)
<label  for = "payment" > Scheduled Paymet date </label>
<input  type = 'text'
        name = 'paymet date'
        formControlName = 'date' >.
</div >.
 <button  type = 'submit'  [disabled] = "!myform. valid"
    pay  </button >.

<div *ngIf = "!thisyform. controlls ['amount']. valid >
   invalid  input
</div >.
<div *ngIf = "!myform. Controls ['bankno']. valid >
   Invalid  Acount  NO
</div >.
<div *ngIf = ' ! myform  Controls ['payment']. valid >
   invalid  Date
</div >
```

```
export class  payment Component {.
myform : FormGoup }

Constructor ( private  datservice : DataService ,
              private  route : ActivatedRoute ;
              private  formBuilder  FormBuilder ) {

this.myform = formBuilder. group ( {
 'amount" : [' ', [validators. required ]]",
 'bankNo' : ['', [Validators. required,
                  this. myValidators ] ],

'date' : [' ', [validators. required ]]

});

OnPayment ( )  {
this. dataService. makePayment (
 this. myform. value. amount,
 this. my form. value. bankNO,
 this. myform  value. payment date );

 this . my form . reset ();

my Validation ( control ; FormControl ) : { [s : string] : boo}
  const id ;     const message.
 this. route. parms. Subscribe ( params => {
      this. id = params ['id']
   })
this. dataservice . get AccountNumber ( id ).
Subscribe ( (data) => {
       this. message = data ;
     }
 );
  return  message ;
}
```

initial payload

1) yes, we can

2) Constructor is called during Construction of Component tree but ngOnInit are called during change detection phase.

3) Components are directives with their Own template but directives are behaviours that are added to existing elements or components

4)
a component is component. It is created with inline template and inline style but no file is created.

5) It set a class called valid to the Component Dom element.

6) None, emulated Native

7) Events, Timer, XHR (application state change) goes through every Component in the Component tree.

8) coz they optimize our Code, which make it fast to execute.

9) We can use OnPush Detection strategy.

10) To get any information about the route associated with loaded component. e.g url, params query params,

11) They create FormGroup instance & name it ngForm & return ngSubmit as Output

12) - It Overcomes CSRF / Highly Sclable / secure and Work very well in stateless state [RestFul API]

13) preboot creates hidden div / browser receive initial payload & user see server view client view is rendered to hidden div. Then preboot switch client view & server views.

* In AOT we only send Empty HTML & JS code but in server side we send rendered HTML to browser