WEEK 4- ANGULAR ASSIGNMENT

1. Forex Calculation Service (ex convert given INR to USD, and vice versa. Able to bring in feature to handle multiple currency conversion features)

app.component.html

```
<div class="container">
    <h1>Forexconversion</h1>
    <div>
      <p>
        <strong>From : </strong>
        <select  [(ngModel)]="From">
          <option *ngFor="let cnt of currency" value={{cnt.id}}>{{cnt.code}}</option>
        </select>

        <strong> To : </strong>
        <select  [(ngModel)]="To">
          <option *ngFor="let cnt of currency">{{cnt.code}}</option>
        </select>
      </p>
    </div>

    <div>
      <p>
         Enter the amount: <input type="number" placeholder="Enter amount" [(ngModel)]="v
alue"/>
      </p>
      <p>
        <button (click)="caluclate()" class="btn btn-primary" >Caluclate</button>
      </p>
    </div>

     <p>The amount in {{To}} is : <strong>{{Result | currency:code}}</strong>  </p>
  </div>
```

app.component.ts

```
import { Component, OnInit} from '@angular/core';
import {ForexserviceService} from './forexservice.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
```

```
  From:number;
  To:string;
  value:number;
  Result:any;
  code:string;
  title = 'angularAssignment1';
  currency=[
    {id:0,code:'AUD'},
    {id:1,code:'CAD'},
    {id:2,code:'EUR'},
    {id:3,code:'GBP'},
    {id:4,code:'INR'},
    {id:5,code:'NZD'},
    {id:6,code:'USD'},

  ]

  constructor(private n:ForexserviceService){

  }
  ngOnInit(): void
  {
    this.n.value_to_be_send_to_components.
    subscribe(
      data=>{
        this.Result=data;
      }
    )

    this.n.value_to_be_send_to_pipecoversion.
    subscribe(
      val=>{
        this.code=val;
      }
    )
  }
  caluclate(){
      this.n.conversion(this.From,this.To,this.value);
  }
}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import {ForexserviceService} from './forexservice.service';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
```

```
import { ChildComponent } from './child/child.component';


@NgModule({
  declarations: [
    AppComponent,
    ChildComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [ForexserviceService],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

ForexserviceService.ts

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ForexserviceService {

   private sendingcaluclateresult=new Subject<number>();
   value_to_be_send_to_components=this.sendingcaluclateresult.asObservable();

   private Pipevalue=new Subject<string>();
   value_to_be_send_to_pipecoversion=this.Pipevalue.asObservable();

   currency_codes=[
   {AUD:1,CAD:0.96,EUR:0.61,GBP:0.56,INR:52.62,NZD:1.08,USD:0.72},      //AUD-0(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}
   {AUD:1.05,CAD:1,EUR:0.64,GBP:0.59,INR:55.06,NZD:1.13,USD:0.75},      //CAD-1(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}
   {AUD:1.63,CAD:1.56,EUR:1,GBP:0.91,INR:85.96,NZD:1.77,USD:1.17},      //EUR-2(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}
   {AUD:1.79,CAD:1.71,EUR:1.09,GBP:1,INR:94.02,NZD:1.94,USD:1.29},      //GBP-3(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}
   {AUD:0.019,CAD:0.018,EUR:0.012,GBP:0.011,INR:1,NZD:0.021,USD:0.014}, //INR-4(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}
   {AUD:0.92,CAD:0.88,EUR:0.57,GBP:0.52,INR:48.56,NZD:1,USD:0.66},      //NZD-5(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}
```

```
   {AUD:1.39,CAD:1.333,EUR:0.85,GBP:0.78,INR:73.32,NZD:1.51,USD:1}];    //USD-6(Id) to {A
UD,CAD,EUR,GBP,INR,NZD,USD}

  constructor() { }

  conversion(FROM:number,TO:string,VALUE:number)
  {
    this.sendingcaluclateresult.next(this.currency_codes[FROM][TO]*VALUE);
    this.Pipevalue.next(TO);
  }

}
```

2. Payment Payload Service – Generates a JSON payload for initiating a payment Request. The JSON Payload should contain :- id, Customer id, payment amount, currency, from Account #, To Account # , beneficiary account , bank charges.

app.component.html

```
<div class="container">
  <h1>Payment section</h1>
  <br>
 <form [formGroup]="myform"  (ngSubmit)="assign();">

   <div class="form-group row">
     <label for="input1" class="col-sm-2 col-form-label">User Name</label>
     <div class="col-sm-10">
       <input type="text" class="form-control" placeholder="User Name" id="input1" formCo
ntrolName="UserName">
       <small *ngIf="myform.get('UserName').touched && myform.get('UserName').hasError('r
equired')">Enter the Username</small>
     </div>
   </div>

   <div class="form-group row">
     <label for="input2" class="col-sm-2 col-form-label">Account No</label>
     <div class="col-sm-10">
       <input type="text" class="form-control" placeholder="Your Account no" id="input2"
formControlName="Account_No_From">
       <small *ngIf="myform.get('Account_No_From').touched && myform.get('Account_No_From
').hasError('required')">Enter your Account No</small>
     </div>
   </div>

   <div class="form-group row">
     <label class="col-sm-2 col-form-label">Account No</label>
     <div class="col-sm-10">
       <input type="text" class="form-control"  placeholder="Beneficiary Account no" form
ControlName="Account_No_To">
```

```
      <small *ngIf="myform.get('Account_No_To').touched && myform.get('Account_No_To').h
asError('required')">Enter the beneficiary Account No</small>
    </div>
  </div>

  <div class="form-group row">
       <label class="col-sm-2 col-label-form">Curreny</label>
       <div class="col-sm-10">
          <select  class="custom-select" formControlName="curr">
              <option value="" disabled>Choose the currency</option>
              <option *ngFor="let cnt of currency" value={{cnt.code}} >{{cnt.name}}</opt
ion>
          </select>
          <small *ngIf="myform.get('curr').touched && myform.get('curr').hasError('requi
red')">Please select the currency</small>
       </div>
  </div>

  <div class="form-group row">
    <label class="col-sm-2 col-label-form">Amount </label>
    <div class="col-sm-10">
      <input type="text" class="form-control" class="form-control" formControlName="amou
nt" >
      <small *ngIf="myform.get('amount').touched && myform.get('amount').hasError('requi
red')">Enter the amount</small>
    </div>
  </div>
  <br>

  <div class="form-group row">
    <button type="submit" class="btn btn-primary" [disabled]="!myform.valid" data-toggle
="modal" data-target="#exampleModalCenter">
      Pay
    </button>
  </div>
 </form>

 <div class="modal fade" id="exampleModalCenter" data-backdrop="static" tabindex="-1" rol
e="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLongTitle" style="text-align: center;">Y
our Payment Status</h5>
      </div>
      <div class="modal-body">
        <div *ngIf="Insuffiecient_Amount">
          In your account you have insufficient amount
        </div>
        <div *ngIf="Wrong_Id">
```

```
          You have enterd a wrong username.
        </div>
        <div *ngIf="Wrong_Acc_no">
          You have entered a wrong account number;
        </div>
        <div *ngIf="Everything_good">
           Payment is done successfully.
        </div>
      </div>
      <div class="modal-footer">
        <button type="submit" class="btn btn-secondary" (click)="reset()" data-dismiss="
modal">Close</button>
      </div>
    </div>
  </div>
 </div>
</div>
```

app.component.ts

```
import { Component, OnInit} from '@angular/core';
import {Validator,FormBuilder, FormGroup, Validators ,ReactiveFormsModule} from '@angular
/forms';
import {ForexserviceService} from './forexservice.service';
import {PaymentPayLoadService} from './payment-pay-load.service';
import {FormControl} from '@angular/forms';
import {NgForm} from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
  Users:any;
  UserName:string;
  Account_No_From:string;
  Account_No_To:string;
  amount:number=0;
  curr: string;



  Insuffiecient_Amount:boolean=false;
  Wrong_Id:boolean=false;
  Wrong_Acc_no:boolean=false;
  Everything_good:boolean=false;
```

```
  currency=[

    {id:0,code:'AUD',name:"Australian Dollar"},
    {id:1,code:'CAD',name:"Canadian Dollar"},
    {id:2,code:'EUR',name:"Euroes"},
    {id:3,code:'GBP',name:"British Pound"},
    {id:4,code:'INR',name:"Indian Rupee"},
    {id:5,code:'NZD',name:"Newzealand Dollar"},
    {id:6,code:'USD',name:"United states Dollar"}

  ]


  myform:FormGroup;




  constructor(private payment:PaymentPayLoadService,private formbuilder: FormBuilder)
  {
    this.myform=formbuilder.group
    ({
      UserName : new FormControl('',[Validators.required,Validators.pattern('^[a-zA-Z ]*$
')]),
      Account_No_From : new FormControl('',[Validators.required,Validators.maxLength(11),
Validators.minLength(11)]),
      Account_No_To : new FormControl('',[Validators.required,Validators.maxLength(11),Va
lidators.minLength(11)]),
      amount : new FormControl('',[Validators.required]),
      curr : new FormControl('',[Validators.required])
    })
  }



  ngOnInit(): void
  {

     this.payment.getCustomersList().
     subscribe
     (
       data=>
       {
         this.Users=data;
       }
     )

  }
  assign()
  {
```

```
    console.log(this.myform.value.curr)
    for(let user of this.Users)
    {
      if(user.Customer_Id!=this.myform.value.UserName)
      {
        this.Wrong_Id=true;
      }
      else if(user.My_Acc_no!=this.myform.value.Account_No_From || user.Beneficiary_Acc_n
o!=this.myform.value.Account_No_To)
      {
        this.Wrong_Acc_no=true;
      }
      else if(user.Amount<this.myform.value.amount)
      {
        this.Insuffiecient_Amount=true;
      }
      else
      {
        this.Everything_good=true;
      }
    }

  }
  reset()
  {
    this.Wrong_Id=false;
    this.Wrong_Acc_no=false;
    this.Insuffiecient_Amount=false;
    this.Everything_good=false;
  }

}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule,ReactiveFormsModule } from '@angular/forms';
import {ForexserviceService} from './forexservice.service';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChildComponent } from './child/child.component';
import { HttpClientModule} from '@angular/common/http';


@NgModule({
  declarations: [
    AppComponent,
    ChildComponent
```

```
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [ForexserviceService],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

payment-pay-load.service.ts

```
import { Injectable } from '@angular/core';
import {HttpClient, HttpClientModule} from '@angular/common/http'

@Injectable({
  providedIn: 'root'
})
export class PaymentPayLoadService {

  constructor(private httpclient:HttpClient) { }

  getCustomersList()
  {
    return this.httpclient.get('http://localhost:3000/Srinivas');
  }

}
```

JSON.json

```
{
  "Sai Lakshmi";
  [
       {
           "Id":1,
           "Customer_Id":"Sai Lakshmi",
           "Amount":10000,
           "My_Acc_no":999999999999,
           "Beneficiary_Acc_no":5555555555;
       }
  ]}
```

3. Check Fraudulent Payment Service - Develop a Service that will detect the fraudulent payment request. You may need to talk to external service to get the list of banned currencies, amount limit,Blocked or banned accounts, date & Time range etc

app.component.html

```
<div class="container">
  <h1>Payment section</h1>
  <br>
 <form [formGroup]="myform"  (ngSubmit)="assign();">

   <div class="form-group row">
     <label for="input1" class="col-sm-2 col-form-label">User Name</label>
     <div class="col-sm-10">
       <input type="text" class="form-control" placeholder="User Name" id="input1" formCo
ntrolName="UserName">
       <small *ngIf="myform.get('UserName').touched && myform.get('UserName').hasError('r
equired')">Enter the Username</small>
     </div>
   </div>

   <div class="form-group row">
     <label for="input2" class="col-sm-2 col-form-label">Account From</label>
     <div class="col-sm-10">
       <input type="text" class="form-control" placeholder="Your Account no" id="input2"
formControlName="Account_No_From">
       <small *ngIf="myform.get('Account_No_From').touched && myform.get('Account_No_From
').hasError('required')">Enter your Account No</small>
     </div>
   </div>

   <div class="form-group row">
     <label class="col-sm-2 col-form-label">Account To</label>
     <div class="col-sm-10">
       <input type="text" class="form-control"  placeholder="Beneficiary Account no" form
ControlName="Account_No_To">
       <small *ngIf="myform.get('Account_No_To').touched && myform.get('Account_No_To').h
asError('required')">Enter the beneficiary Account No</small>
     </div>
   </div>

   <div class="form-group row">
        <label class="col-sm-2 col-label-form">Curreny</label>
        <div class="col-sm-10">
          <select  class="custom-select" formControlName="curr">
              <option value="" disabled>Choose the currency</option>
              <option *ngFor="let cnt of currency" value={{cnt.code}} >{{cnt.name}}</opt
ion>
          </select>
          <small *ngIf="myform.get('curr').touched && myform.get('curr').hasError('requi
red')">Please select the currency</small>
```

```
        </div>
    </div>

    <div class="form-group row">
      <label class="col-sm-2 col-label-form">Amount </label>
      <div class="col-sm-10">
        <input type="text" class="form-control" class="form-control" formControlName="amou
nt" >
        <small *ngIf="myform.get('amount').touched && myform.get('amount').hasError('requi
red')">Enter the amount</small>
      </div>
    </div>
    <br>

    <div class="form-group row">
      <button type="submit" class="btn btn-primary" [disabled]="!myform.valid" data-toggle
="modal" data-target="#exampleModalCenter">
        Pay
      </button>
    </div>
  </form>

  <div class="modal fade" id="exampleModalCenter" data-backdrop="static" tabindex="-1" rol
e="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLongTitle" style="text-align: center;">Y
our Payment Status</h5>
        </div>
        <div class="modal-body">
          <div *ngIf="Insuffiecient_Amount">
                {{Insuffiecient_msg}}
          </div>
          <div *ngIf="exceedAmount">
                {{ExceedAmount_msg}}
          </div>
          <div *ngIf="banned_ac">
                {{banned_ac_msg}}.
          </div>
          <div *ngIf="banned_curr">
                {{banned_curr_msg}}
          </div>
          <div *ngIf="Wrong_Id">
                {{  wrong_Id_msg}}
          </div>
          <div *ngIf="Wrong_Acc_no">
                {{wrong_Acc_msg}}
          </div>
          <div *ngIf="Everything_good">
```

```
                {{success_msg}}
        </div>
      </div>
      <div class="modal-footer">
        <button type="submit" class="btn btn-secondary" (click)="reset()" data-dismiss="
modal">Close</button>
      </div>
    </div>
  </div>
 </div>
</div>
```

app.component.ts

```
import { Component, OnInit} from '@angular/core';
import {Validator,FormBuilder, FormGroup, Validators ,ReactiveFormsModule} from '@angular
/forms';
import {ForexserviceService} from './forexservice.service';
import {PaymentPayLoadService} from './payment-pay-load.service';
import {FormControl} from '@angular/forms';
import {NgForm} from '@angular/forms';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
  Users:any;
  UserName:string;
  Account_No_From:string;
  Account_No_To:string;
  amount:number=0;
  curr: string;



  Insuffiecient_Amount:boolean=false;
  Wrong_Id:boolean=false;
  Wrong_Acc_no:boolean=false;
  Everything_good:boolean=false;

  currency=[

    {id:0,code:'AUD',name:"Australian Dollar"},
    {id:1,code:'CAD',name:"Canadian Dollar"},
    {id:2,code:'EUR',name:"Euroes"},
    {id:3,code:'GBP',name:"British Pound"},
    {id:4,code:'INR',name:"Indian Rupee"},
```

```
    {id:5,code:'NZD',name:"Newzealand Dollar"},
    {id:6,code:'USD',name:"United states Dollar"}

  ]


  myform:FormGroup;




  constructor(private payment:PaymentPayLoadService,private formbuilder: FormBuilder)
  {
    this.myform=formbuilder.group
    ({
      UserName : new FormControl('',[Validators.required,Validators.pattern('^[a-zA-Z ]*$
')]),
      Account_No_From : new FormControl('',[Validators.required,Validators.maxLength(11),
Validators.minLength(11)]),
      Account_No_To : new FormControl('',[Validators.required,Validators.maxLength(11),Va
lidators.minLength(11)]),
      amount : new FormControl('',[Validators.required]),
      curr : new FormControl('',[Validators.required])
    })
  }



  ngOnInit(): void
  {

     this.payment.getCustomersList().
     subscribe
     (
       data=>
       {
         this.Users=data;
       }
     )

  }
  assign()
  {
    console.log(this.myform.value.curr)
    for(let user of this.Users)
    {
      if(user.Customer_Id!=this.myform.value.UserName)
      {
        this.Wrong_Id=true;
      }
```

```
      else if(user.My_Acc_no!=this.myform.value.Account_No_From || user.Beneficiary_Acc_n
o!=this.myform.value.Account_No_To)
      {
        this.Wrong_Acc_no=true;
      }
      else if(user.Amount<this.myform.value.amount)
      {
        this.Insuffiecient_Amount=true;
      }
      else
      {
        this.Everything_good=true;
      }
    }

  }
  reset()
  {
    this.Wrong_Id=false;
    this.Wrong_Acc_no=false;
    this.Insuffiecient_Amount=false;
    this.Everything_good=false;
  }

}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule,ReactiveFormsModule } from '@angular/forms';
import {ForexserviceService} from './forexservice.service';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChildComponent } from './child/child.component';
import { HttpClientModule} from '@angular/common/http';


@NgModule({
  declarations: [
    AppComponent,
    ChildComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
```

```
  providers: [ForexserviceService],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

FradulentPaymentService.ts

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

@Injectable({
  providedIn: 'root'
})
export class FraudulentPaymentService {

  constructor(private httpclient :HttpClient) { }

  getCustomersList()
  {
    return this.httpclient.get('http://localhost:3000/Customer1');
  }

  getBannedCurrencyList()
  {
    return this.httpclient.get('http://localhost:3000/BannedCurrencies');
  }

  getBannedAccountList()
  {
    return this.httpclient.get('http://localhost:3000/BannedAccounts')
  }
}
```

FradulentList.json

```
{
  "Customer1":
  [
      {
          "Id":1,
           "Customer_Id":"Sai Lakshmi",
           "Amount":100000,
           "My_Acc_no":"99999999999",
           "Beneficiary_Acc_no":"444444444444"
      }
  ],
```

```
  "BannedCurrencies":

      {
          "curr1": "HEC",
          "curr2":"MCC",
          "curr3":"JIC",
          "curr4":"LID"
      },

  "BannedAccounts":

      {
          "acc1": "11111111111",
          "acc2": "12121212121",
          "acc3": "88888888888",
          "acc4": "21212121212"
      }

}
```

4. UnitConversionService - Develop a service that performs conversion of one unit to another unit. Ex, conversion of cm to mtr, centigrade to Fahrenheit etc

app.component.html

```
<div class="button">
  <button class="btn btn-primary" style="margin: 5px;" (click)="D1()" data-toggle="button
"> Length</button>
  <button class="btn btn-primary" style="margin: 5px;" (click)="D2()" data-toggle="button
">Temperature</button>
</div>
<div *ngIf="!Display1">

  <div class="container">
    <h1>OneConversion in Length</h1>
    <div>
      <p>
        <strong>From : </strong>
        <select   [(ngModel)]="From1">
          <option    *ngFor="let v of values1" value={{v.id}}>{{v.name}}</option>
        </select>

        <strong> To : </strong>
        <select   [(ngModel)]="To1">
          <option  *ngFor="let v of values1" value={{v.code}}>{{v.name}}</option>
        </select>
      </p>
    </div>
```

```
    <div>
      <p>
         Enter the value: <input type="number" placeholder="Enter value" [(ngModel)]="val
1"/>
      </p>
      <p>
        <button (click)="ans1()" class="btn btn-primary" >Caluclate</button>
      </p>
    </div>

     <p>The value in {{To1}} is : <strong>{{Result}}</strong>  </p>
  </div>

</div>

<div *ngIf="!Display2">
  <div class="container">
    <h1>OneConversion In Temperature</h1>
    <div>
      <p>
        <strong>From2 : </strong>
        <select  [(ngModel)]="From2">
          <option *ngFor="let v of values2" value={{v.id}}>{{v.name}}</option>
        </select>

        <strong> To2 : </strong>
        <select  [(ngModel)]="To2">
          <option *ngFor="let v of values2" value={{v.code}}>{{v.name}}</option>
        </select>
      </p>
    </div>

    <div>
      <p>
         Enter the value: <input type="number" placeholder="Enter value" [(ngModel)]="val
2"/>
      </p>
      <p>
        <button (click)="ans2()" class="btn btn-primary" >Caluclate</button>
      </p>
    </div>

     <p>The value in {{To2}} is : <strong>{{Result}}</strong>  </p>
  </div></div>
```

app.component.ts

```
import { asNativeElements, Component, OnInit} from '@angular/core';
import {Validator,FormBuilder, FormGroup, Validators ,ReactiveFormsModule} from '@angular
/forms';
import {ForexserviceService} from './forexservice.service';
```

```
import {PaymentPayLoadService} from './payment-pay-load.service';
import { FraudulentPaymentService } from './fraudulent-payment.service';
import {FormControl} from '@angular/forms';
import {NgForm} from '@angular/forms';

import {OneconversionService} from './oneconversion.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
  From1:number;
  To1:string;
  val1:number;

  From2:string;
  To2:string;
  val2:number

  Result:any;

  Display1:boolean=false;
  Display2:boolean=true;

  values1=
  [
    {id:0,code:'mts',name:'Meters'},
    {id:1,code:'kms',name:'Kilometers'},
    {id:2,code:'cms',name:'Centimeters'},
    {id:3,code:'millims',name:'Millimeter'},
    {id:4,code:'microms',name:'Micrometers'},
    {id:5,code:'nanms',name:'Nanometers'},
    {id:6,code:'mile',name:'Mile'},
    {id:7,code:'yard',name:'Yard'},
    {id:8,code:'foot',name:'Foot'},
    {id:9,code:'Inch',name:'Inch'}
  ]

  values2=
  [
    {id:'Farenheit',code:'Farenheit',name:'Farenheit'},
    {id:'Celsius',code:'Celsius',name:'Celsius'},
    {id:'Kelvin',code:'Kelvin',name:'Kelvin'}
  ]

  constructor(private n:OneconversionService){}
  ngOnInit(): void
```

```
  {
     this.n.value_to_be_send_to_components.
     subscribe
     (
       data=>{
                this.Result=data;
              }
     )
     if(this.Result<1 && this.Result>0)
     {
        this.Result=this.Result.toExponential();
     }
  }
  ans1()
  {
   this.n.caluclate2(this.From1,this.To1,this.val1);
  }

  ans2()
  {
    this.n.caluclate1(this.From2,this.To2,this.val2);
  }

  D1()
  {
    this.Display1=false;
    this.Display2=true;
    this.Result=null
  }
  D2()
  {
    this.Display1=true;
    this.Display2=false;
    this.Result=null;
  }

}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule,ReactiveFormsModule } from '@angular/forms';
import {ForexserviceService} from './forexservice.service';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChildComponent } from './child/child.component';
import { HttpClientModule} from '@angular/common/http';
import {OneconversionService} from './oneconversion.service';
```

```
@NgModule({
  declarations: [
    AppComponent,
    ChildComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [ForexserviceService,
              OneconversionService],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

OneConversionService.ts

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs'

@Injectable({
  providedIn: 'root'
})
export class OneconversionService {

  private sendingcaluclateresult=new Subject<number>();
  value_to_be_send_to_components=this.sendingcaluclateresult.asObservable()

  conversion=
  [
    {mts:1,kms:0.001,cms:100,millims:1000,microms:10**6,nanms:10**9,mile:6.21371*(10**-4)
,yard:1.09361,foot:3.28084,Inch:39.3701},
    {mts:1000,kms:1,cms:10**5,millims:10**6,microms:10**9,nanms:10**12,mile:6.21371*(10**
-1),yard:1093.61,foot:3280.84,Inch:39370.1},
    {mts:0.01,kms:10**-5,cms:1,millims:10,microms:10**4,nanms:10**7,mile:6.213*(10**-6),y
ard:0.0109361,foot:0.0328084,Inch:0.393701},
    {mts:0.001,kms:10**-6,cms:0.1,millims:1,microms:1000,nanms:10**6,mile:6.213*(10**-7),
yard:0.00109361,foot:0.00328089,Inch:0.0393701},
    {mts:10**-6,kms:10**-9,cms:10**-4,millims:0.001,microms:1,nanms:1000,mile:6.2137*(10*
*-10),yard:1.0936*(10**-6),foot:3.280*(10**-6),Inch:3.937*(10**-5)},
    {mts:10**-9,kms:10**-12,cms:10**-7,millims:10**-6,microms:0.001,nanms:1,mile:6.2137*(
10**-13),yard:1.0936*(10**-9),foot:3.2808*(10**-9),Inch:3.937*(10**-8)},
    {mts:1609.34,kms:1.6093,cms:160934,millims:1.609*(10**6),microms:1.609*(10**9),nanms:
1.609*(10**12),mile:1,yard:1760,foot:5280,Inch:63360},
    {mts:0.9144,kms:0.0009144,cms:91.44,millims:914.4,microms:914400,nanms:9.144*(10**8),
mile:5.6812*(10**-4),yard:1,foot:3,Inch:36},
```

```
    {mts:0.3048,kms:3.048*(10**-4),cms:30.48,millims:304.8,microms:304800,nanms:3.048*(10
**8),mile:1.893*(10**-4),yard:0.333,foot:1,Inch:12},
    {mts:0.0254,kms:2.54*(10**-5),cms:2.54,millims:25.4,microms:25400,nanms:2.54*(10**7),
mile:1.573*(10**-5),yard:0.0277,foot:0.00833,Inch:1}
  ]
  caluclate1(from:string,to:string,value:number)
  {
    if(to==='Celsius' && from==='Farenheit')
    {
      this.sendingcaluclateresult.next(this.F_to_c(value))
    }
    else if(to==='Farenheit' &&  from==='Celsius')
    {
      this.sendingcaluclateresult.next(this.C_to_f(value))
    }
    else if(to==='Kelvin' && from==='Celsius')
    {
      this.sendingcaluclateresult.next(this.C_to_k(value))
    }
    else if(to==='Celsius' && from==='Kelvin')
    {
      this.sendingcaluclateresult.next(this.K_to_c(value))
    }
    else if(to==='Kelvin' && from==='Farenheit')
    {
      this.sendingcaluclateresult.next(this.F_to_k(value))
    }
    else if(to==='Farenheit' && from==='Kelvin')
    {
      this.sendingcaluclateresult.next(this.K_to_f(value))
    }
    else
    {
      this.sendingcaluclateresult.next(value)
    }

  }

  F_to_c(f:number)
  {
      return (f-32)/1.8;
  }
  C_to_f(c:number)
  {
      return (c*1.8)+32;
  }
  C_to_k(f:number)
  {
    return f+273.15;
  }
```

```
  K_to_c(f:number)
  {
    return f-273.15;
  }
  F_to_k(f:number)
  {
    return ((f-32)*(5/9)+273.15);
  }
  K_to_f(f:number)
  {
    return ((f-273.15)*(9/5)+32)
  }

  caluclate2(from:number,to:string,value:number)
  {

    this.sendingcaluclateresult.next(this.conversion[from][to]*value)

  }
}
```

5. Develop Pipes to calculate age from the DOB, format Credit Card Number, Display Currency Conversion value , Display available balance for the given account, Display available credit limit for the given credit card

app.component.html

```
<div class="container">

  <div class="row">
     <label class="col-sm-2">DOB:</label>
     <input type="date" style="margin-left: 8px;" [(ngModel)]="birthdate">
     <div style="margin-left: 8px;">
         <strong>{{birthdate | ageCaluclator: birthdate }} </strong>
     </div>
  </div>

  <br>

  <div class="row">
      <label class="col-sm-2"> CC_number:</label>
      <input type="text" [(ngModel)]="num" placeholder="CreditCard number">
      <div style="margin: 8px;">
         <strong> {{num | formattingCreditCardNumber:num}}</strong>
      </div>
  </div>

  <br>

  <div class="row">
```

```
    <label class="col-sm-2">Account number:</label>
    <input type="text" [(ngModel)]="Account_balance" placeholder="Account number">
    <div>
       <strong> {{Account_balance | showbalance:Account_balance}}</strong>
    </div>
  </div>


  <br>

  <div class="row">
    <label class="col-sm-2">creditcard number:</label>
    <input type="text"  id="idnum" onkeyup="addHyphen(this)" [(ngModel)]="Limit" placehol
der="CreditCard number">
    <div>
       <strong> {{Limit | showbalanceofcreditcardnumber:Limit}}</strong>
    </div>
  </div>

  <br>

  <div>
      <h5>Currency conversion</h5>
      <div>
          <p>
            <strong>From : </strong>
            <select  [(ngModel)]="From">
              <option *ngFor="let cnt of currency" value={{cnt.id}}>{{cnt.code}}</option>
            </select>

            <strong> To : </strong>
            <select  [(ngModel)]="To">
              <option *ngFor="let cnt of currency">{{cnt.code}}</option>
            </select>
          </p>
      </div>

      <div>
          <p>
             Enter the amount: <input type="number" placeholder="Enter amount" [(ngModel)
]="value"/>
          </p>
      </div>

      <p><strong>The value in {{To}} is {{value | currencyconverter:From:To:value | curre
ncy:To}}</strong>  </p>
  </div>

</div>
```

app.component.ts

```
import { asNativeElements, Component, OnInit} from '@angular/core';
import {Validator,FormBuilder, FormGroup, Validators ,ReactiveFormsModule} from '@angular
/forms';
import {ForexserviceService} from './forexservice.service';
import {PaymentPayLoadService} from './payment-pay-load.service';
import { FraudulentPaymentService } from './fraudulent-payment.service';
import {FormControl} from '@angular/forms';
import {NgForm} from '@angular/forms';

import {OneconversionService} from './oneconversion.service';

import {Pipe,PipeTransform} from '@angular/core'


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
   ngOnInit(): void {

   }


   age:number;
   birthdate:string
   num:string;

   Account_balance:string
   Limit:string;

   From:number;
  To:string;
  value:number;
  Result:any;
  code:string;

  currency=[
    {id:0,code:'AUD'},
    {id:1,code:'CAD'},
    {id:2,code:'EUR'},
    {id:3,code:'GBP'},
    {id:4,code:'INR'},
    {id:5,code:'NZD'},
    {id:6,code:'USD'},
```

```
  ]
}
```

c.js

```
function addHyphen (element)
{
  let ele = document.getElementById(element.id);
  ele = ele.value.split('-').join('');
  let finalVal = ele.match(/.{1,4}/g).join('-');
  document.getElementById(element.id).value = finalVal;
}
```

age-calculator.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'ageCaluclator'
})
export class AgeCaluclatorPipe implements PipeTransform {

  transform(value: string, bir:string): string
  {
    if(bir!=undefined)
    {
      var   today= new Date();
      const b =new Date(bir)
      var diff_year=today.getFullYear()-b.getFullYear();
      var diff_month=today.getMonth()-b.getMonth();
      if(diff_month<0 || (diff_month===0 && today.getDate()<b.getDate()))
      {
         diff_year--;
      }
      return `Your  age is ${diff_year}`
    }
  }
}
```

formatting-creditcard-number.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'formattingCreditCardNumber'
})
export class FormattingCreditCardNumberPipe implements PipeTransform {
```

```
  transform(value: string,n:string): string
  {
    if(n!=undefined)
    {
       var v = value.replace(/\s+/g, '').replace(/[^0-9]/gi, '')
       var matches = value.match(/\d{4,16}/g);
       var match = matches && matches[0] || ''
       var parts = []
       for (var i=0, len=match.length; i<len; i+=4) {
            parts.push(match.substring(i, i+4))
       }
       if (parts.length)
        {
           return   `Formatted credit card number ${parts.join('  ')}`
        }
       else
      {
           return `Formateed credit card number ${value}`;
       }
    }
   }

}
```

showbalance.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'showbalance'
})
export class ShowbalancePipe implements PipeTransform {

  transform(value: string): string
  {
    if(value!=undefined)
    {

      if(value==='11111111111')
      {
        return 'Your balance is $20,000'
      }
    }
  }


}
```

showbalanceofcreditcardnumber.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'showbalanceofcreditcardnumber'
})
export class ShowbalanceofcreditcardnumberPipe implements PipeTransform {

  transform(value:string): string
  {
    console.log(value)
    if(value!=undefined)
    {

      if(value.length<19)
      {
        return ' '
      }
      else if(value==="8888-8888-8888-8888")
      {
        return 'you can withdraw 10,000 per day';
      }
      else{
        return 'You enterd a invalid credt card number'
      }

    }
  }

}
```

currencyconversion.pipe.ts

```
import { Pipe, PipeTransform } from '@angular/core';
import { from } from 'rxjs';

@Pipe({
  name: 'currencyconverter'
})
export class CurrencyconverterPipe implements PipeTransform {

  transform(value:number,From:string,To:number): number
  {
    if(From!=undefined && To!=undefined && value!=undefined)
    {
      var currency_codes=[
        {AUD:1,CAD:0.96,EUR:0.61,GBP:0.56,INR:52.62,NZD:1.08,USD:0.72},      //AUD-0(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}
        {AUD:1.05,CAD:1,EUR:0.64,GBP:0.59,INR:55.06,NZD:1.13,USD:0.75},      //CAD-1(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}
```

```
        {AUD:1.63,CAD:1.56,EUR:1,GBP:0.91,INR:85.96,NZD:1.77,USD:1.17},       //EUR-2(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}
        {AUD:1.79,CAD:1.71,EUR:1.09,GBP:1,INR:94.02,NZD:1.94,USD:1.29},       //GBP-3(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}
        {AUD:0.019,CAD:0.018,EUR:0.012,GBP:0.011,INR:1,NZD:0.021,USD:0.014}, //INR-4(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}
        {AUD:0.92,CAD:0.88,EUR:0.57,GBP:0.52,INR:48.56,NZD:1,USD:0.66},       //NZD-5(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}
        {AUD:1.39,CAD:1.333,EUR:0.85,GBP:0.78,INR:73.32,NZD:1.51,USD:1}];     //USD-6(Id)
to {AUD,CAD,EUR,GBP,INR,NZD,USD}

        var res=currency_codes[From][To]*value;
        return res

    }
  }


}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule,ReactiveFormsModule } from '@angular/forms';
import {ForexserviceService} from './forexservice.service';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChildComponent } from './child/child.component';
import { HttpClientModule} from '@angular/common/http';
import {OneconversionService} from './oneconversion.service';
import { AgeCaluclatorPipe } from './age-caluclator.pipe';
import { FormattingCreditCardNumberPipe } from './formatting-credit-card-number.pipe';
import { CurrencyconverterPipe } from './currencyconverter.pipe';
import { ShowbalancePipe } from './showbalance.pipe';
import { ShowbalanceofcreditcardnumberPipe } from './showbalanceofcreditcardnumber.pipe';


@NgModule({
  declarations: [
    AppComponent,
    ChildComponent,
    AgeCaluclatorPipe,
    FormattingCreditCardNumberPipe,
    CurrencyconverterPipe,
    ShowbalancePipe,
    ShowbalanceofcreditcardnumberPipe
  ],
  imports: [
    BrowserModule,
```

```
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [ForexserviceService,
              OneconversionService],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

6. Connect to back end api over Promise and fetch following details (required for the drop downs) – Customer Types, CreditCard Types, List of Currencies, List of Countries. List of Cities (for the given country id)

app.component.html

```
<div class="container">

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select Customer:</label>
      <div class="col-sm-10">
         <select  class="custom-select">
             <option value="" >Choose the customer type</option>
             <option *ngFor="let cnt of customerTypes">{{cnt.name}}</option>
         </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select currency:</label>
      <div class="col-sm-10">
         <select  class="custom-select">
             <option value="" >Choose the currency</option>
             <option *ngFor="let cnt of currency">{{cnt.code}}</option>
         </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select Credit Card Type:</label>
      <div class="col-sm-10">
         <select  class="custom-select">
             <option value="" >Choose the credit type</option>
             <option *ngFor="let cnt of  creditcardTypes">{{cnt.name}}</option>
         </select>
      </div>
  </div>
```

```
  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select Country:</label>
      <div class="col-sm-10">
         <select  class="custom-select" (change)="changeCountry($event.target.value)">
             <option value="" >Choose the currency</option>
             <option *ngFor="let cnt of countryList" value={{cnt.id}} >{{cnt.name}}</opti
on>
         </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select City:</label>
      <div class="col-sm-10">
         <select  class="custom-select">
             <option value="" >Choose the city</option>
             <option *ngFor="let cnt of cities">{{cnt}}</option>
         </select>
      </div>
  </div>

</div>
```

app.component.ts

```
import { asNativeElements, Component, OnInit} from '@angular/core';
import {Validator,FormBuilder, FormGroup, Validators ,ReactiveFormsModule} from '@angular
/forms';
import {ForexserviceService} from './forexservice.service';
import {PaymentPayLoadService} from './payment-pay-load.service';
import { FraudulentPaymentService } from './fraudulent-payment.service';
import {FormControl} from '@angular/forms';
import {NgForm} from '@angular/forms';

import {OneconversionService} from './oneconversion.service';

import {Pipe,PipeTransform} from '@angular/core'


@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit
{
   countryList:any;
   customerTypes:any;
   creditcardTypes:any;
   currency:any;
```

```
  constructor(private n:ForexserviceService){}
  ngOnInit(): void
  {
       this.n.getcountry().
       toPromise().then
       (

         data1=>
         {
           this.countryList=data1;
         }
       )

       this.n.getcustomer().
       toPromise().then
       (
          data2=>
          {
            this.customerTypes=data2;
          }
       )

       this.n.getcreditcard().
       toPromise().then
       (
         data3=>
         {
           this.creditcardTypes=data3;
         }
       )

       this.n.getcurrency().
       toPromise().then
       (
         data4=>
         {
           this.currency=data4;
         }
       )
  }

 cities: Array<any>;
 changeCountry(count) {
   this.cities = this.countryList.find(con => con.id == count).cities;
 }
}
```

dropdown.service.ts

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

@Injectable({
  providedIn: 'root'
})
export class ForexserviceService
{

    constructor(private httpclient:HttpClient){}

    getcountry()
    {
      return this.httpclient.get('http://localhost:3000/countryList')

    }

    getcustomer()
    {
      return this.httpclient.get('http://localhost:3000/Customer_types')
    }

    getcreditcard()
    {
      return this.httpclient.get('http://localhost:3000/Crdeitcard_types')

    }

    getcurrency()
    {
      return this.httpclient.get('http://localhost:3000/currency')
    }
}
```

JSON.json

```
{
  "countryList": [
      { "id":0,"name": "Germany", "cities": ["Duesseldorf", "Leinfelden-Echterdingen", "E
schborn"] },
      { "id":1,"name": "Spain", "cities": ["Barcelona","Dech","Eroti","Uysi","Kripo"] },
      { "id":2,"name": "USA", "cities": ["Downers Grove","San Francico","Godi","Lodi"] },
      { "id":3,"name": "Mexico","cities": ["Puebla","Merio","Giodi","Luop"] },
      { "id":4,"name": "China", "cities": ["Beijing","Hiuwai","Godiya","Gyma","Kyyu"] },
      { "id":5,"name":"India", "cities": ["Rajahmundry","Vizag","Vijawada","Kakinada","Gu
ntur"]}
    ],
```

```
    "Customer_types": [
      {"id":0,"name":"Single Account"},
      {"id":1,"name":"Multiple Acount"},
      {"id":2,"name":"Joint Account"}
    ],

    "Crdeitcard_types":[
        {"name":"Visa"},
        {"name":"Master card"},
        {"name":"American Express"},
        {"name":"SBI platinum"},
        {"name":"citi Bank card"}
    ],

    "currency": [
      {"code":"AUD"},
      {"code":"CAD"},
      {"code":"EUR"},
      {"code":"GBP"},
      {"code":"INR"},
      {"code":"NZD"},
      {"code":"USD"}

    ]
}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule,ReactiveFormsModule } from '@angular/forms';
import {ForexserviceService} from './forexservice.service';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ChildComponent } from './child/child.component';
import { HttpClientModule} from '@angular/common/http';
import {OneconversionService} from './oneconversion.service';
import { AgeCaluclatorPipe } from './age-caluclator.pipe';
import { FormattingCreditCardNumberPipe } from './formatting-credit-card-number.pipe';
import { CurrencyconverterPipe } from './currencyconverter.pipe';
import { ShowbalancePipe } from './showbalance.pipe';
import { ShowbalanceofcreditcardnumberPipe } from './showbalanceofcreditcardnumber.pipe';


@NgModule({
  declarations: [
    AppComponent,
    ChildComponent,
```

```
    AgeCaluclatorPipe,
    FormattingCreditCardNumberPipe,
    CurrencyconverterPipe,
    ShowbalancePipe,
    ShowbalanceofcreditcardnumberPipe
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ReactiveFormsModule,
    HttpClientModule
  ],
  providers: [ForexserviceService,
              OneconversionService],

  bootstrap: [AppComponent]
})
export class AppModule { }
```

7. Connect to back end api over observable for all :- Fetch list of Currencies, and using Switch Map find and remove unsupported or banned currencies, Fetch the matching transaction list based on the user search text box. Fetch the list of Application Status, and using pipe & map filter function remove in correct or partially submitted documents.

app.component.html

```
<div class="container">

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select Customer:</label>
      <div class="col-sm-10">
         <select  class="custom-select">
             <option value="" >Choose the customer type</option>
             <option *ngFor="let cnt of customerTypes">{{cnt.name}}</option>
         </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select currency:</label>
      <div class="col-sm-10">
```

```
        <select  class="custom-select">
            <option value="" >Choose the currency</option>
            <option *ngFor="let cnt of currency">{{cnt.code}}</option>
        </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select Credit Card Type:</label>
      <div class="col-sm-10">
        <select  class="custom-select">
            <option value="" >Choose the credit type</option>
            <option *ngFor="let cnt of  creditcardTypes">{{cnt.name}}</option>
        </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select Country:</label>
      <div class="col-sm-10">
        <select  class="custom-select" (change)="changeCountry($event.target.value)">
            <option value="" >Choose the currency</option>
            <option *ngFor="let cnt of countryList" value={{cnt.id}} >{{cnt.name}}</opti
on>
        </select>
      </div>
  </div>

  <div class="form-group row">
      <label class="col-sm-2 col-label-form">Select City:</label>
      <div class="col-sm-10">
        <select  class="custom-select">
            <option value="" >Choose the city</option>
            <option *ngFor="let cnt of cities">{{cnt}}</option>
        </select>
      </div>
  </div>

</div>
```

app.component.ts

```
import { AfterViewInit, asNativeElements, Component, OnInit, ViewChild} from '@angular/co
re';
import {Validator,FormBuilder, FormGroup, Validators ,ReactiveFormsModule} from '@angular
/forms';
import {ForexserviceService} from './forexservice.service';
import {PaymentPayLoadService} from './payment-pay-load.service';
import { FraudulentPaymentService } from './fraudulent-payment.service';
import {FormControl} from '@angular/forms';
```

```
import {NgForm} from '@angular/forms';

import {OneconversionService} from './oneconversion.service';

import {Pipe,PipeTransform} from '@angular/core'

import {filter,map,debounceTime,distinctUntilChanged, switchMap} from 'rxjs/operators';
import { Observable } from 'rxjs';
import { Currency } from './Currency';
import { search } from './search.interface';
import { documents} from './Docuements';




@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit,AfterViewInit
{


  currencies: any;
  SearchResults:search;
  count:number;
  doc:any

  constructor(private dataService: ForexserviceService) { }

  public curr :Observable<Currency[]>=null;
  public docc :Observable<documents[]>=null;

  ngOnInit()
  {
    this.dataService.getcurrencies().pipe
    (
    map((data:Currency[])=>
    {
      let crr:Currency[]=[]
      data.map(d=>
        {
         if(d.name!='AAA' && d.name!='HHH') { crr.push(d) };
        }
      )
      return crr;
    }))
    .subscribe
    (
```

```
      (res:Currency[])=>
       {
         this.currencies=res;
       }
    )

    this.dataService.getdocuements().pipe
    (
    map((data:documents[])=>
    {
      let crr:documents[]=[]
      data.map(d=>
        {
         if(d.status!='Partially submitted' && d.status!='Incorrect') { crr.push(d) };
        }
      )
      return crr;
    }))
    .subscribe
    (
      (res:documents[])=>
       {
         this.doc=res;
       }
    )




   }

  @ViewChild('searchForm') searchForm:NgForm

  ngAfterViewInit():void
  {


      const formvalue=this.searchForm.valueChanges;

      formvalue.
      pipe(filter(()=>this.searchForm.valid),map(data=>data.searchTerm),debounceTime(500)
,
              distinctUntilChanged(),switchMap(data=>this.dataService.getsearches(data)))
.
      subscribe(res=>{this.SearchResults=res; this.count=Object.keys(res).length})
  }

}
```

observable.service.ts

```
import { HttpClient,HttpClientModule } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable, of, Subject } from 'rxjs';
import { search } from './search.interface';

@Injectable({
  providedIn: 'root'
})

export class ForexserviceService
{

  constructor(private httpclient:HttpClient){}

   url='http://localhost:3000/Transactions';

   getsearches(searchTerm):Observable<search>
   {
       return this.httpclient.get<search>(this.url+'?q='+searchTerm);
   }

  getcurrencies()
  {
    return this.httpclient.get('http://localhost:3000/Currencies')
  }

  getdocuements()
  {
    return this.httpclient.get('http://localhost:3000/docuements')
  }
}
```

currency.ts

```
export class Currency
{
    name:string;
}
```

search.ts

```
export interface search
  {
      Id:string;
      Transaction_No:string;
      To:string;
      Amount:string;
```

```
  }
```

documents.ts

```
export class documents
{
    name:string;
    status:string;
}
```

JSON.json

```
{
  "Currencies":
  [
      {"name":"INR"},
      {"name":"GBP"},
      {"name":"NZD"},
      {"name":"USD"},
      {"name":"HHH"},
      {"name":"EUR"},
      {"name":"CAD"},
      {"name":"AUD"},
      {"name":"AAA"}
  ],

  "Transactions":
  [
      {
          "Id":"Sai Lakshmi Chekuri",
          "Transaction_No":"TRN001223334",
          "To":"To the City bank for shopping purpose",
          "Amount":"$1234"
      },
      {
          "Id":"Divya",
          "Transaction_No":"TRN001223534",
          "To":"To the China bank for shopping purpose",
          "Amount":"$1234"
      },
      {
          "Id":"Kiran",
          "Transaction_No":"TRN001220334",
          "To":"To the baroda bank for hdfc purpose",
          "Amount":"$1234"
      },
      {
          "Id":"Abhijeet",
```

```
        "Transaction_No":"TRN001223334",
        "To":"To the Mrui bank for home loan purpose",
        "Amount":"$1234"
    },
    {
        "Id":"venkatesh",
        "Transaction_No":"TRN001228904",
        "To":"To the KUNGFU bank for loan purpose",
        "Amount":"$123904"
    },
    {
        "Id":"Mehaboob",
        "Transaction_No":"TRN0012893334",
        "To":"To the Priya bank for loan purpose",
        "Amount":"$1234"
    },
    {
        "Id":"Saranya",
        "Transaction_No":"TRN0019223334",
        "To":"To the cyma bank for home loan purpose",
        "Amount":"$1234"
    },
    {
        "Id":"Ranjith meha",
        "Transaction_No":"TRN0012673334",
        "To":"To the higu bank for shopping purpose",
        "Amonut":"$19034"
    },
    {
        "Id":"Sriniu koyu",
        "Transaction_No":"TRN001923334",
        "To":"To the GIdI bank for cinema loan purpose",
        "Amonut":"$6784"
    },
    {
        "Id":"Harry Potter",
        "Transaction_No":"TRN001563334",
        "To":"To the mufi bank for loan of house purpose",
        "Amonut":"$67789"
    },
    {
        "Id":"Harish Mandela",
        "Transaction_No":"TRN001223334",
        "To":"To the Hiwai bank for House loan purpose",
        "Amonut":"$8904"
    },
    {
        "Id":"Heema Konidhi",
        "Transaction_No":"TRN001229034",
        "To":"To the ZERA bank for visa loan purpose",
```

```
        "Amonut":"$123478"
    },
    {
        "Id":"Meena Agarthala",
        "Transaction_No":"TRN001223904",
        "To":"To the MIF bank for student loan purpose",
        "Amonut":"$127834"
    },
    {
        "Id":"Jimmy cheyu",
        "Transaction_No":"TRN0012456334",
        "To":"To the Kia bank for loan purpose",
        "Amonut":"$123004"
    },
    {
        "Id":"Henny Kena",
        "Transaction_No":"TRN004563334",
        "To":"To the Mardhad bank for loan purpose",
        "Amonut":"$123400"
    },
    {
        "Id":"John Kenny",
        "Transaction_No":"TRN00128034",
        "To":"To the HDFC bank for building purpose",
        "Amonut":"$12304"
    },
    {
        "Id":"Jyothi ",
        "Transaction_No":"TRN004224334",
        "To":"To the SBI bank for soung purpose",
        "Amonut":"$123094"
    },
    {
        "Id":"Paul",
        "Transaction_No":"TRN001228934",
        "To":"To the hiuo bank for himing purpose",
        "Amount":"$123400"
    }
],
"documents":
[
    {
        "name":"John",
        "status":"submitted"
    },
    {
        "name":"Mark",
        "status":"submitted"
    },
    {
```

```
            "name":"Beny",
            "status":"Partially submitted"
        },
        {
            "name":"John",
            "status":"Incorrect"
        }
    ]
}
```