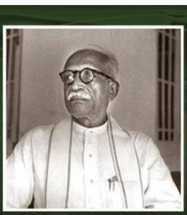


Sprint-2 Plan

Employee Detail Management

1. Establishing Project Folder Architecture
2. Server App Initialization
3. **Sprint-2 Execution** [2(a), 2(b), 2(c), **2(d)**(i)(ii)(iii)**(iv)**(v), 2(e), 2(f)]
4. Completing Sprint-1 Objectives

2(a) Employee Module - Analysis, Design & DB-Preparation
2(b) Employee View
2(c) Employee Search

2(d) Employee Insert

2(e) Employee Update
2(f) Employee Delete

==

2(d) Employee Insert

1. Folder and Project preparation with Backups
2. Prepare Supportive Data for the Insert Form
3. Prepare Regex(Regular Expression) Service for the Insert Form

4. **Client App**
    1. Define Form Controls with Required Validator
    2. Define HTML Form Controls with relevant Binding
    3. **Test for Error Messages**
    4. Test for Confirmation Message
    5. Service Implementation with POST Request

5. Server-App

==

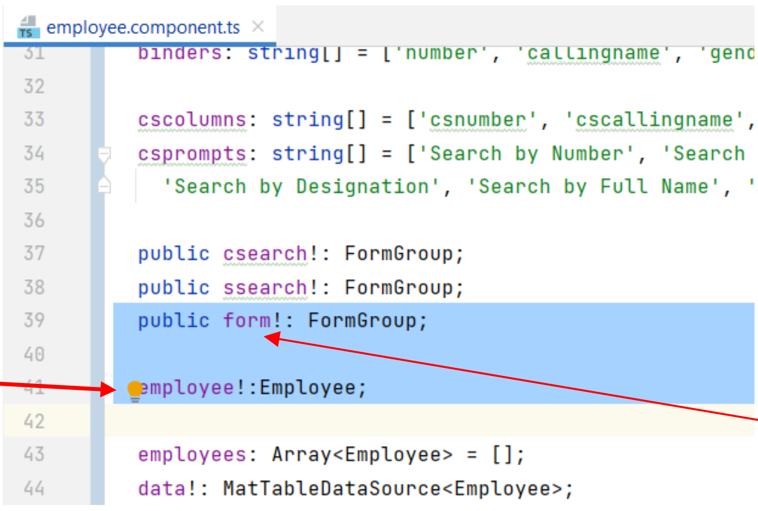
Modification to Last Version

1. **Test for Error Message**

Modification to Last Version

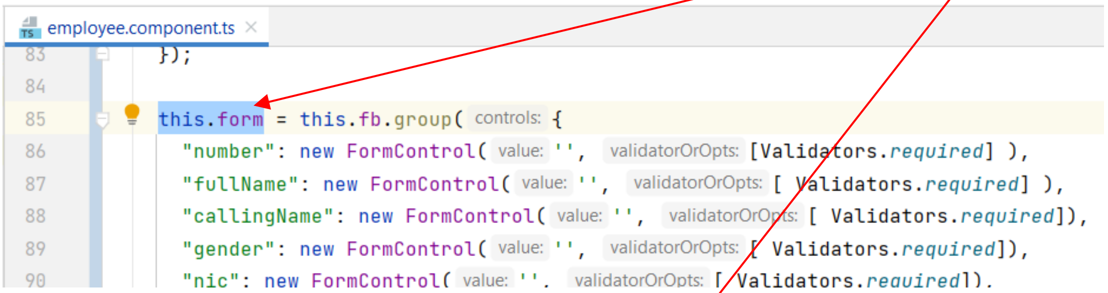
Rename Add Form from “employee” to “form”

Add global variable as “employee” of type Employee to refer bind data from the form.



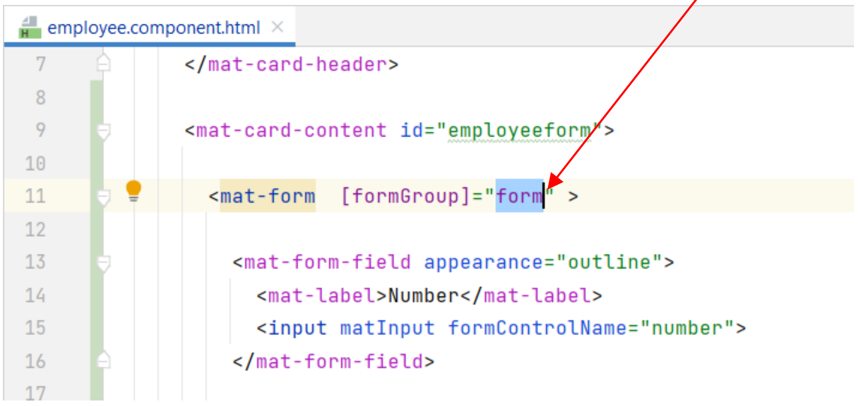
employee.component.ts

```
binders: string[] = ['number', 'callingname', 'genc

cscolumns: string[] = ['csnumber', 'cscallingname',
csprompts: string[] = ['Search by Number', 'Search
  'Search by Designation', 'Search by Full Name', '

public csearch!: FormGroup;
public ssearch!: FormGroup;
public form!: FormGroup;

employee!:Employee;

employees: Array<Employee> = [];
data!: MatTableDataSource<Employee>;
```



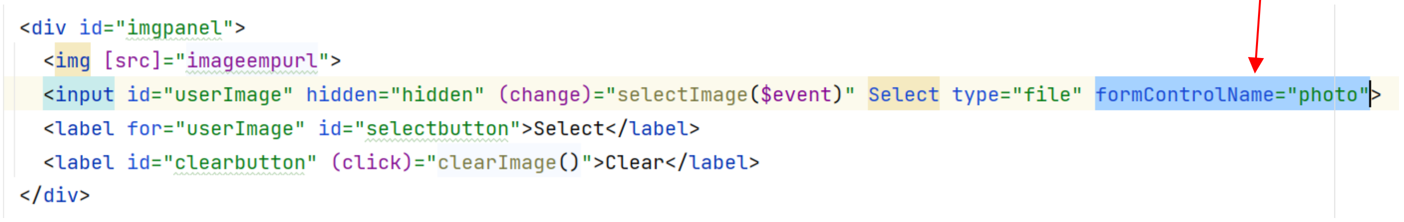
employee.component.ts

```
  });

  this.form = this.fb.group( controls: {
    "number": new FormControl( value: '',  validatorOrOpts: [Validators.required] ),
    "fullName": new FormControl( value: '',  validatorOrOpts: [ Validators.required] ),
    "callingName": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
    "gender": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
    "nic": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
```

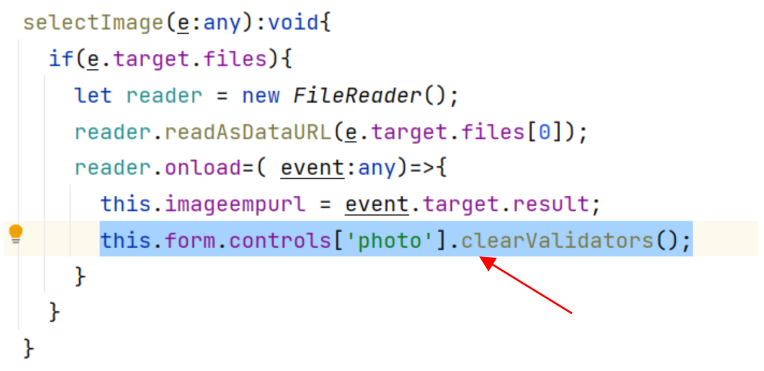


employee.component.html

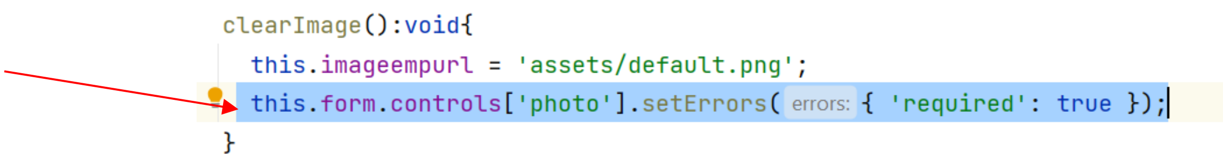
```
  </mat-card-header>

  <mat-card-content id="employeeform">

    <mat-form  [formGroup]="form" >

      <mat-form-field appearance="outline">
        <mat-label>Number</mat-label>
        <input matInput formControlName="number">
      </mat-form-field>
```



```
<div id="imgpanel">
  <img [src]="imageempurl">
  <input id="userImage" hidden="hidden" (change)="selectImage($event)" Select type="file" formControlName="photo">
  <label for="userImage" id="selectbutton">Select</label>
  <label id="clearbutton" (click)="clearImage()">Clear</label>
</div>
```



```
selectImage(e:any):void{
  if(e.target.files){
    let reader = new FileReader();
    reader.readAsDataURL(e.target.files[0]);
    reader.onload=( event:any)=>{
      this.imageempurl = event.target.result;
      this.form.controls['photo'].clearValidators();
    }
  }
}
```

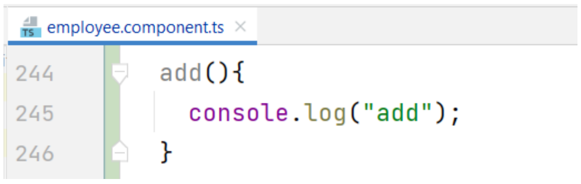


```
clearImage():void{
  this.imageempurl = 'assets/default.png';
  this.form.controls['photo'].setErrors( errors: { 'required': true });
}
```

1. Test for Error Message

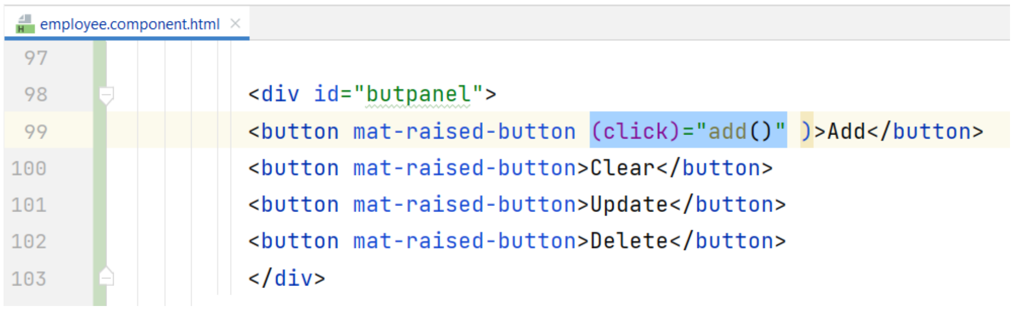
1. **Define the “add()” with “console.log(“add”)” for event handler testing**
2. **Register/Bind event handler with the HTML UI Template and Test**
3. **Define “getErrors()” and call when “add()” called**

1. Define the “add()” with “console.log(“add”)” for event handler testing



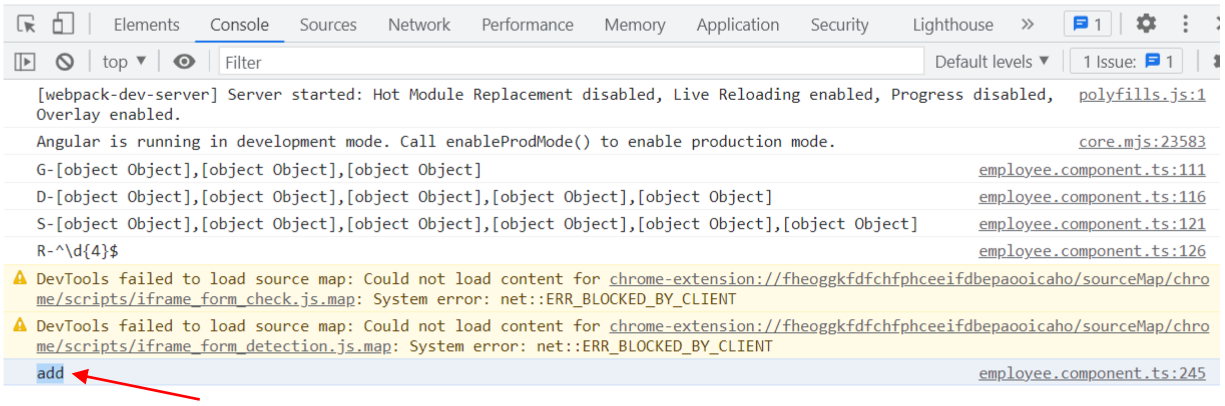
```
add(){
  console.log("add");
}
```

2. Register/Bind event handler with the HTML UI Template and Test



```
<div id="butpanel">
<button mat-raised-button (click)="add()" >Add</button>
<button mat-raised-button>Clear</button>
<button mat-raised-button>Update</button>
<button mat-raised-button>Delete</button>
</div>
```

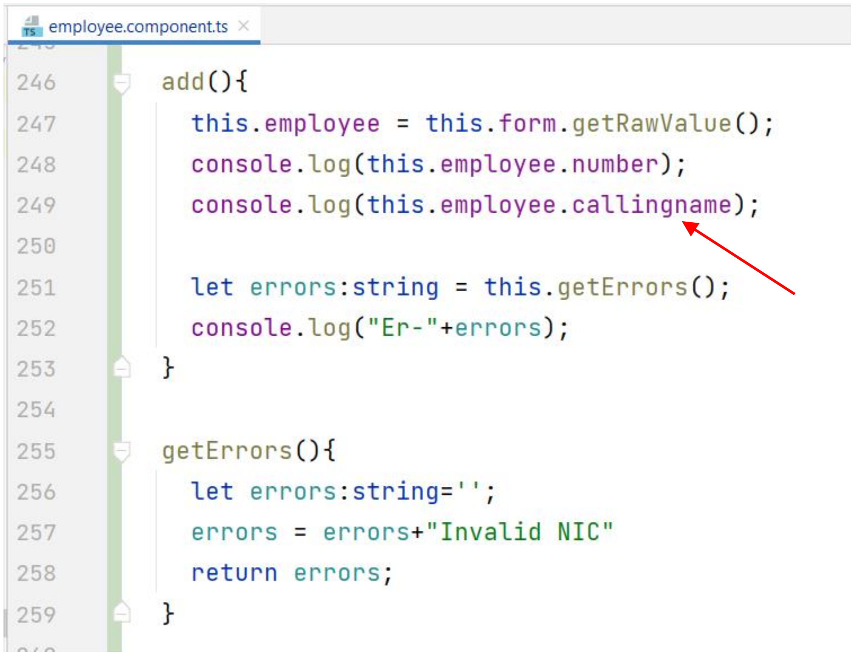
Run the Project, Open Employee Module, Click on Add Button, Observe Console Outputs



```
[webpack-dev-server] Server started: Hot Module Replacement disabled, Live Reloading enabled, Progress disabled,   polyfills.js:1
Overlay enabled.
Angular is running in development mode. Call enableProdMode() to enable production mode.                        core.mjs:23583
G-[object Object],[object Object],[object Object]                                                        employee.component.ts:111
D-[object Object],[object Object],[object Object],[object Object],[object Object]                        employee.component.ts:116
S-[object Object],[object Object],[object Object],[object Object],[object Object],[object Object]        employee.component.ts:121
R-^\d{4}$                                                                                                employee.component.ts:126
⚠ DevTools failed to load source map: Could not load content for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chro
me/scripts/iframe_form_check.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
⚠ DevTools failed to load source map: Could not load content for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chro
me/scripts/iframe_form_detection.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
add                                                                                                      employee.component.ts:245
```

3. Define “getErrors()” and call when “add()” called

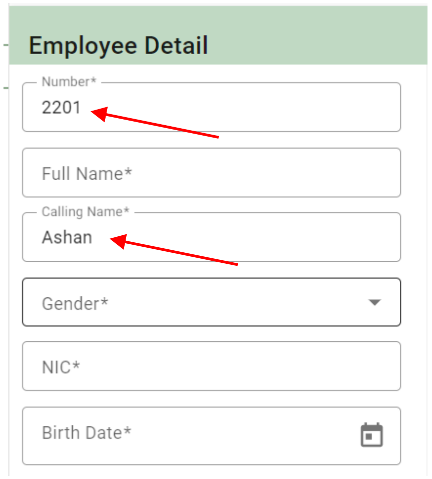
Code as follows,



employee.component.ts

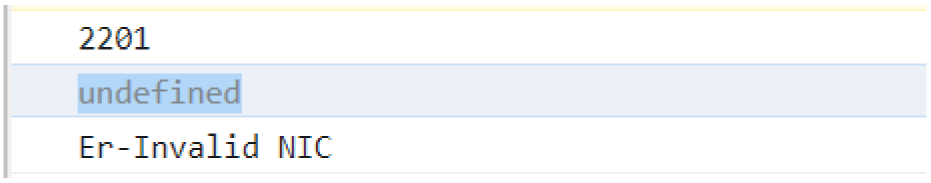
```
add(){
  this.employee = this.form.getRawValue();
  console.log(this.employee.number);
  console.log(this.employee.callingname);

  let errors:string = this.getErrors();
  console.log("Er-"+errors);
}

getErrors(){
  let errors:string='';
  errors = errors+"Invalid NIC"
  return errors;
}
```

Run the Project → Open Employee → Type “Number” – 2201, “Calling Name” – Ashan → Click on Add Button → Observe Console Output for Bind Data (number and callingname) and Returned Error Message “Invalid NIC”



Employee Detail

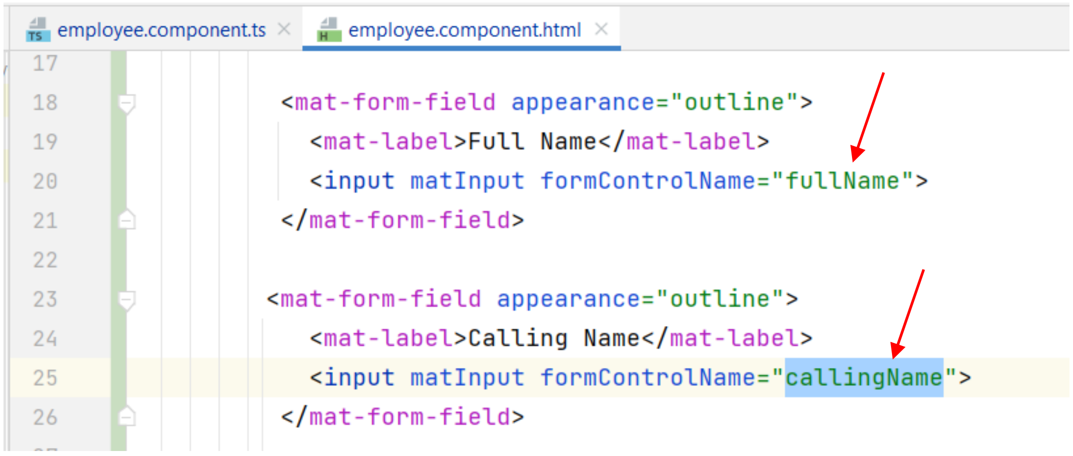
Number*	2201
Full Name*	
Calling Name*	Ashan
Gender*	
NIC*	
Birth Date*	



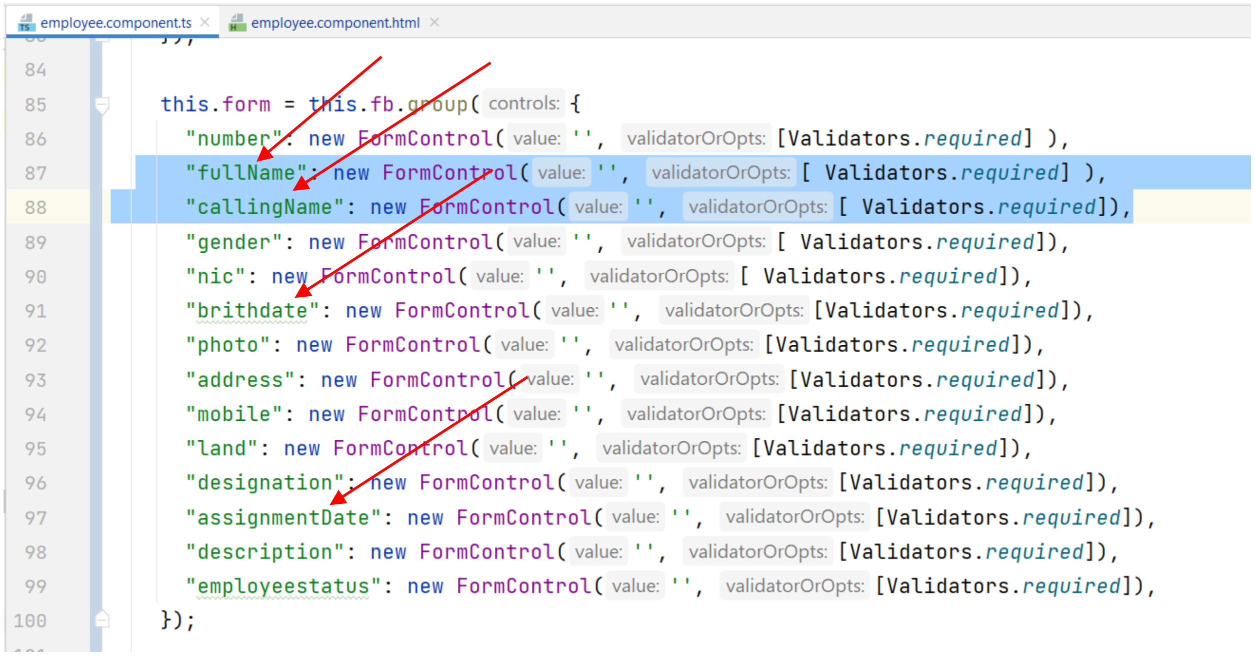
```
2201
undefined
Er-Invalid NIC
```

Why Undefined the “employee.callingname” ?

Form Control Names are not same as Properties of Employee Object → Cant bind

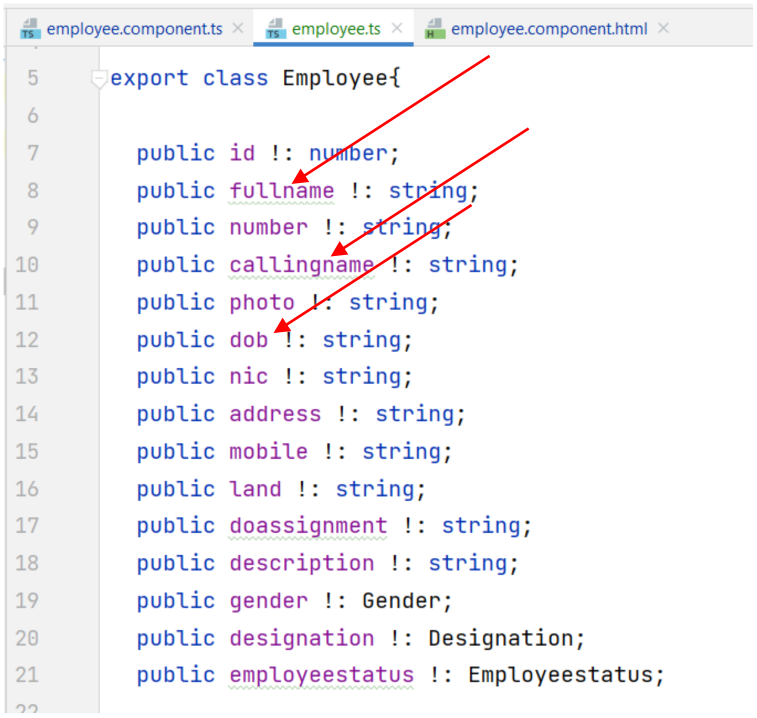

employee.component.ts × employee.component.html ×

```
          <mat-form-field appearance="outline">
            <mat-label>Full Name</mat-label>
            <input matInput formControlName="fullName">
          </mat-form-field>

        <mat-form-field appearance="outline">
            <mat-label>Calling Name</mat-label>
            <input matInput formControlName="callingName">
          </mat-form-field>
```


employee.component.ts × employee.component.html ×

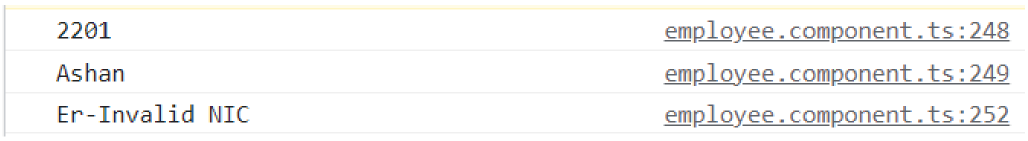
```
    this.form = this.fb.group( controls: {
      "number": new FormControl( value: '',  validatorOrOpts: [Validators.required] ),
      "fullName": new FormControl( value: '',  validatorOrOpts: [ Validators.required] ),
      "callingName": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
      "gender": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
      "nic": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
      "brithdate": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "photo": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "address": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "mobile": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "land": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "designation": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "assignmentDate": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "description": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
      "employeestatus": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
    });
```

Change them "callingName" to "callingname", "fullName" to "fullname"
Look for other differences as well


employee.component.ts × employee.ts × employee.component.html ×

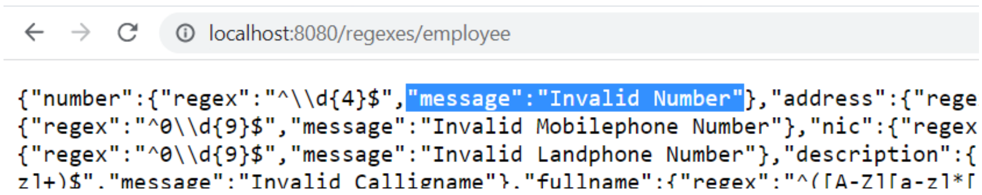
```
export class Employee{

  public id !: number;
  public fullname !: string;
  public number !: string;
  public callingname !: string;
  public photo !: string;
  public dob !: string;
  public nic !: string;
  public address !: string;
  public mobile !: string;
  public land !: string;
  public doassignment !: string;
  public description !: string;
  public gender !: Gender;
  public designation !: Designation;
  public employeestatus !: Employeestatus;
```

Correct "employeecomponnet.ts" and "employeecomponnet.html" to reflect correct names and rerun the test.
(callingName, fullName, birthdate, assignmentDate → fully simple letters)



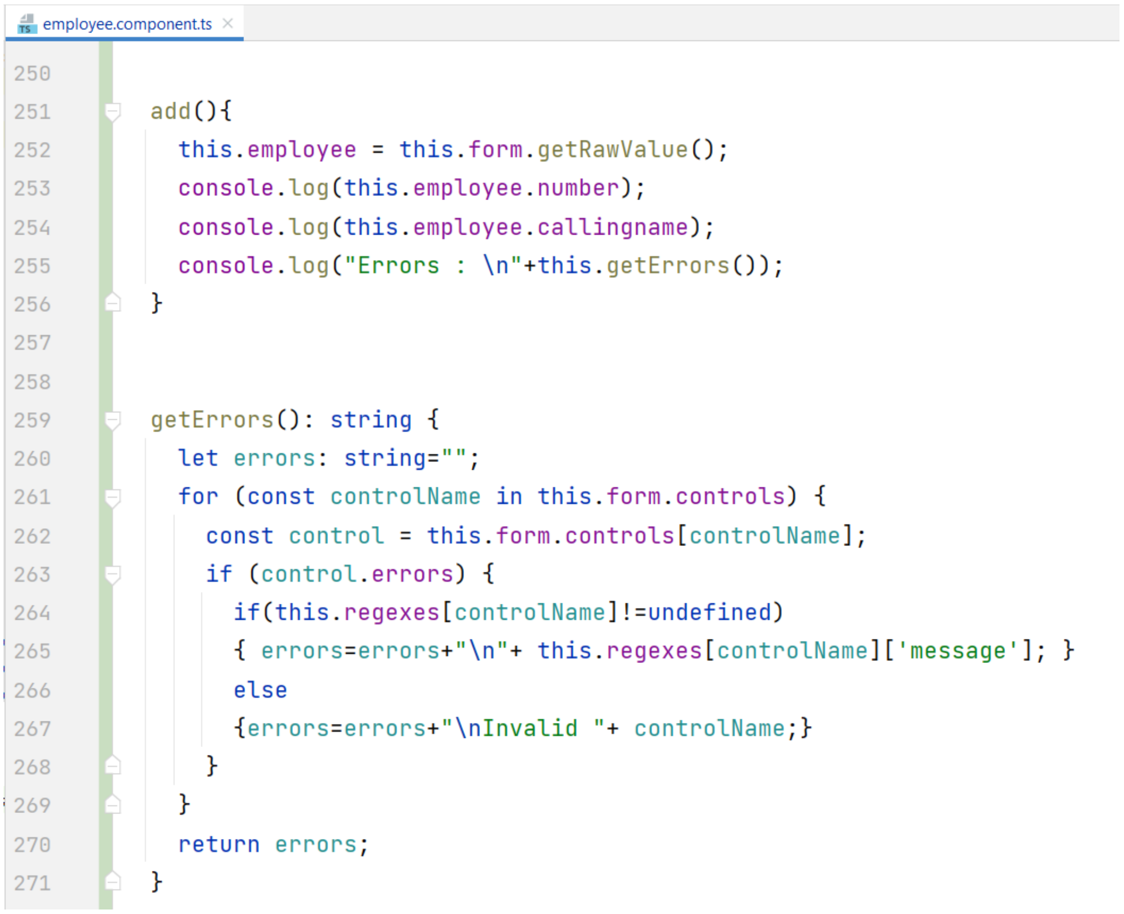
```
2201                                   employee.component.ts:248
Ashan                                  employee.component.ts:249
Er-Invalid NIC                         employee.component.ts:252
```

Complete the "getErrors()" method to find the Errors in the Form based on the given Validators and show Error Messages received from the Server within "regexes"



localhost:8080/regexes/employee

```
{"number":{"regex":"^\\d{4}$","message":"Invalid Number"},"address":{"rege
{"regex":"^0\\d{9}$","message":"Invalid Mobilephone Number"},"nic":{"regex
{"regex":"^0\\d{9}$","message":"Invalid Landphone Number"},"description":{
z]+)$","message":"Invalid Callingname"},"fullname":{"regex":"^([A-Z][a-z]*[
```

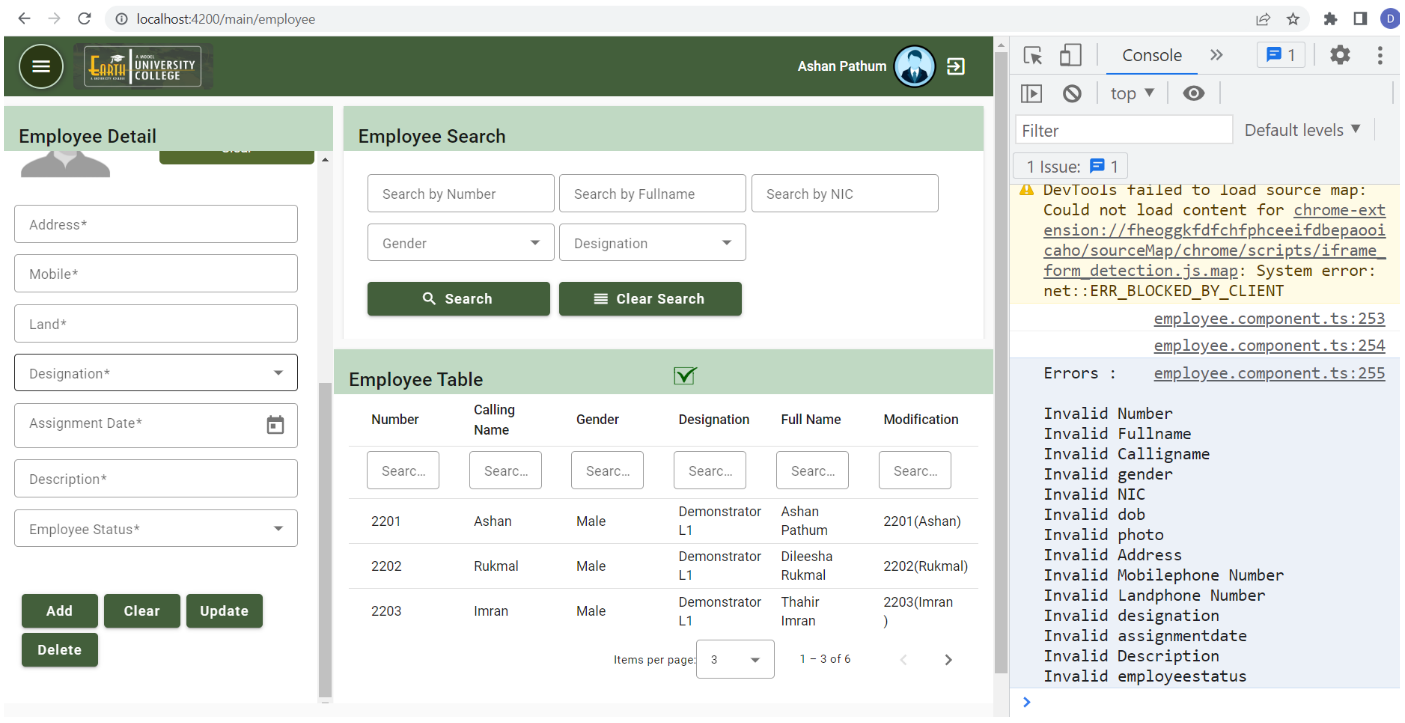


employee.component.ts

```
add(){
  this.employee = this.form.getRawValue();
  console.log(this.employee.number);
  console.log(this.employee.callingname);
  console.log("Errors : \n"+this.getErrors());
}


getErrors(): string {
  let errors: string="";
  for (const controlName in this.form.controls) {
    const control = this.form.controls[controlName];
    if (control.errors) {
      if(this.regexes[controlName]!=undefined)
      { errors=errors+"\n"+ this.regexes[controlName]['message']; }
      else
      {errors=errors+"\nInvalid "+ controlName;}
    }
  }
  return errors;
}
```

Testing (13 Tests for each Form Field)
Re run the Project and Open Employee Form
Click on "Add" button → Observe Console Output
Fill one field by field while clicking on the "Add" button and observing the output

Error Messages will be removed one by one when you insert each
The Pattern is not important as the Validators.required is the only Validator added
We will add Pattern (Regex from Server Regexes) Validation after this testing process



Employee Detail

Address*

Mobile*

Land*

Designation*

Assignment Date*

Description*

Employee Status*

Add Clear Update Delete

Employee Search

Search by Number | Search by Fullname | Search by NIC

Gender | Designation

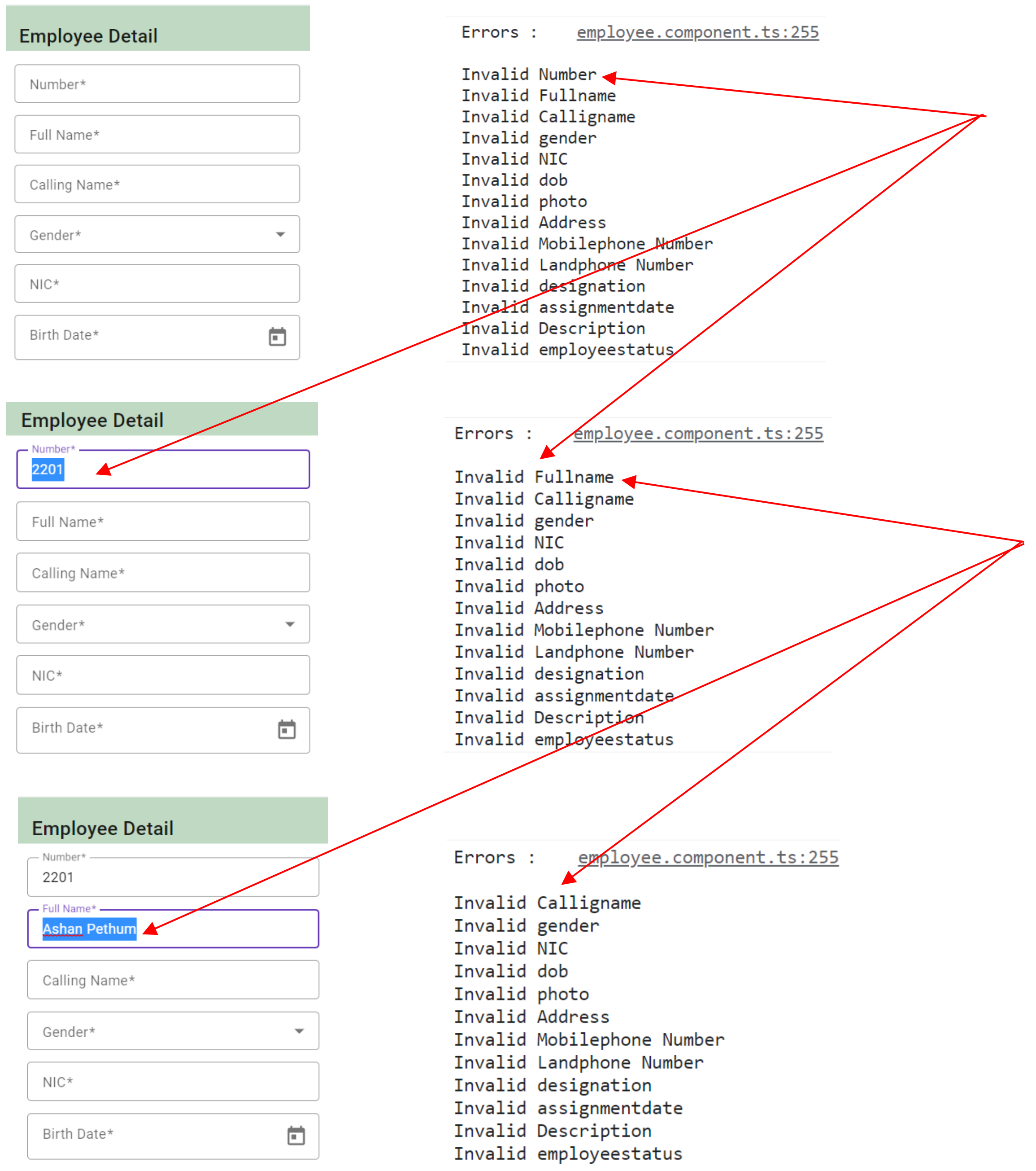
Search | Clear Search

Employee Table

Number	Calling Name	Gender	Designation	Full Name	Modification
Searc...	Searc...	Searc...	Searc...	Searc...	Searc...
2201	Ashan	Male	Demonstrator L1	Ashan Pathum	2201(Ashan)
2202	Rukmal	Male	Demonstrator L1	Dileesha Rukmal	2202(Rukmal)
2203	Imran	Male	Demonstrator L1	Thahir Imran	2203(Imran)

Items per page: 3 1 – 3 of 6

```
DevTools failed to load source map: Could not load content for chrome-extension://fheoggkfdfchfphceeifdbepaooicaho/sourceMap/chrome/scripts/iframe_form_detection.js.map: System error: net::ERR_BLOCKED_BY_CLIENT
                                         employee.component.ts:253
                                         employee.component.ts:254
Errors :                                 employee.component.ts:255

Invalid Number
Invalid Fullname
Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```



Employee Detail

Number*

Full Name*

Calling Name*

Gender*

NIC*

Birth Date*

```
Errors :     employee.component.ts:255

Invalid Number
Invalid Fullname
Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

Employee Detail

Number*: 2201

Full Name*

Calling Name*

Gender*

NIC*

Birth Date*

```
Errors :     employee.component.ts:255

Invalid Fullname
Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

Employee Detail

Number*: 2201

Full Name*: Ashan Pethum

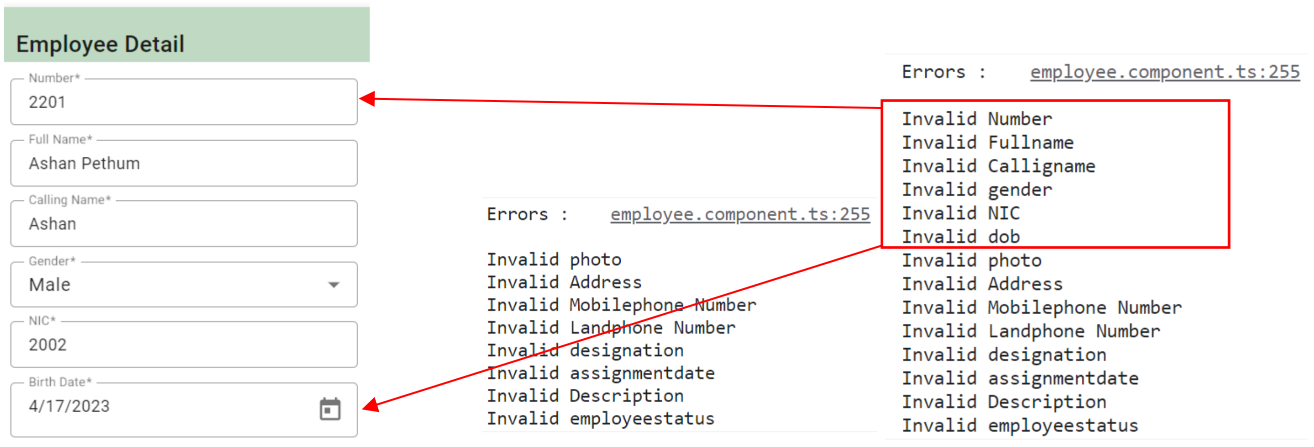
Calling Name*

Gender*

NIC*

Birth Date*

```
Errors :     employee.component.ts:255

Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```



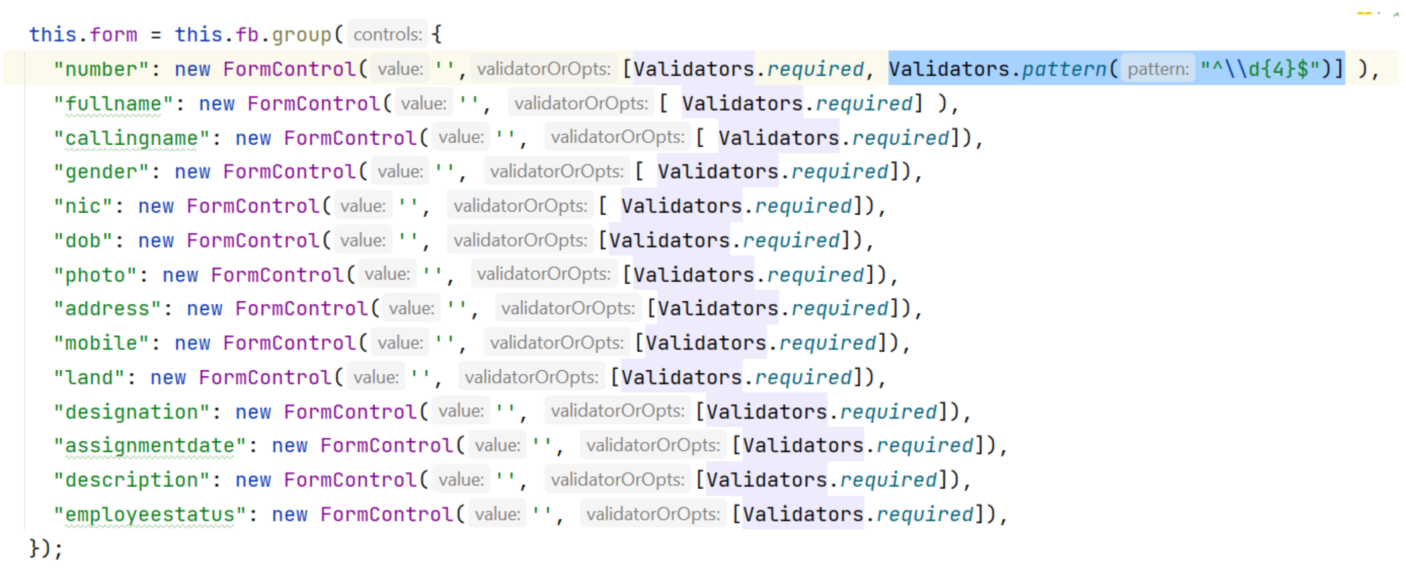
Number* 2201
Full Name* Ashan Pethum
Calling Name* Ashan
Gender* Male
NIC* 2002
Birth Date* 4/17/2023

```
Errors :    employee.component.ts:255

Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

```
Errors :    employee.component.ts:255

Invalid Number
Invalid Fullname
Invalid Callingname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

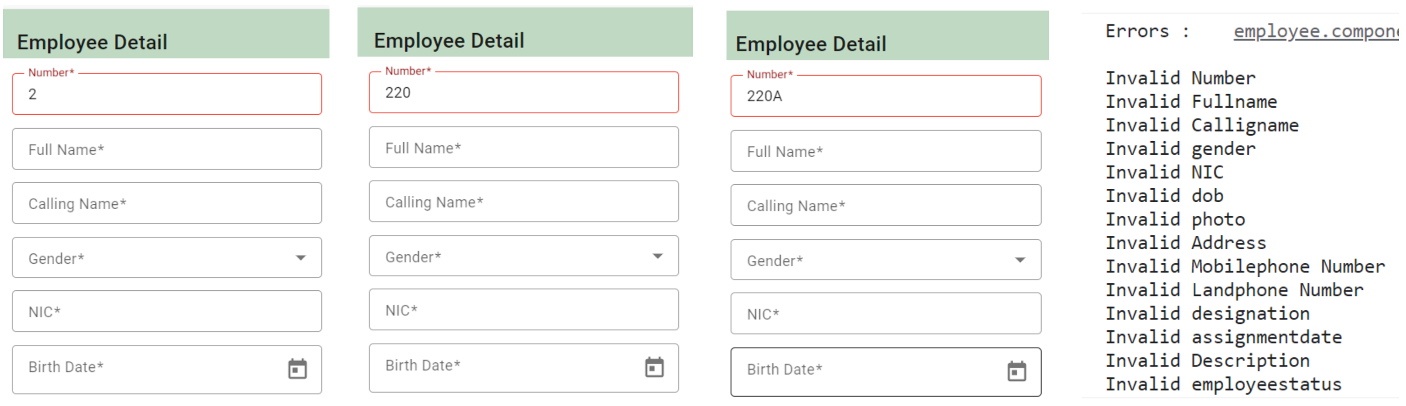
Test the Number with 2, 22, 220A and 2201 ← All Input Values will be validated
Because Validator will only check for Required not for the Pattern

Apply Pattern Validation with Regex for required Form Controllers



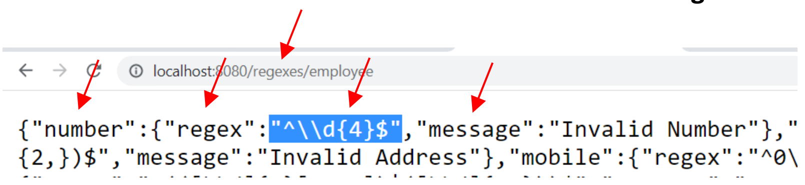
```
this.form = this.fb.group( controls: {
  "number": new FormControl( value: '', validatorOrOpts: [Validators.required, Validators.pattern( pattern: "^\\d{4}$")] ),
  "fullname": new FormControl( value: '',  validatorOrOpts: [ Validators.required] ),
  "callingname": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
  "gender": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
  "nic": new FormControl( value: '',  validatorOrOpts: [ Validators.required]),
  "dob": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "photo": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "address": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "mobile": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "land": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "designation": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "assignmentdate": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "description": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
  "employeestatus": new FormControl( value: '',  validatorOrOpts: [Validators.required]),
});
```

It will check for both required and the Pattern also



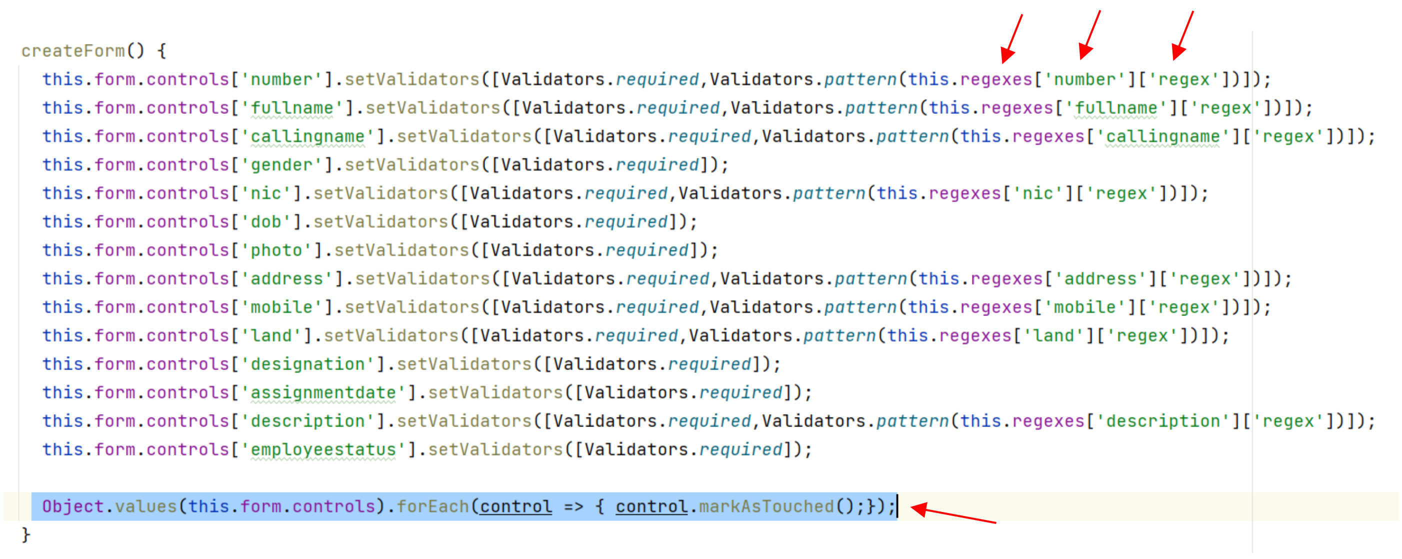
```
Errors :    employee.compon

Invalid Number
Invalid Fullname
Invalid Callingname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

Regex Patterns are requested from the Server and Update when regexes received.
Hence Pattern Validators must be set into the Form Controls once the regexes loaded.

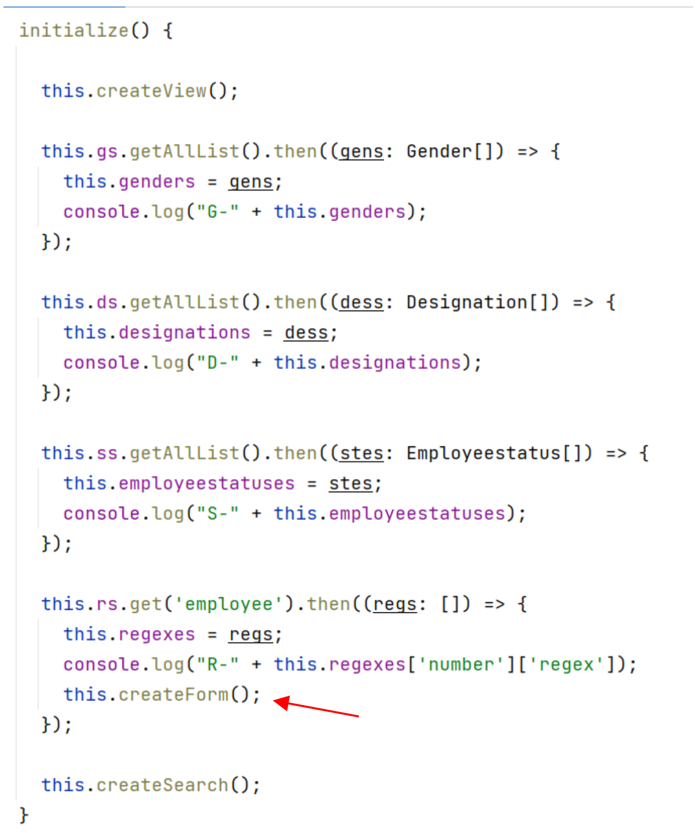


```
{"number":{"regex":"^\\d{4}$","message":"Invalid Number"},"
{2,})$","message":"Invalid Address"},"mobile":{"regex":"^0\
```

Define “createForm()” and called it after regexes received in the then() of the Promise

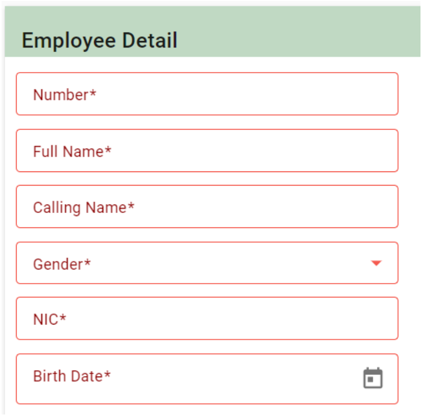


```
createForm() {
  this.form.controls['number'].setValidators([Validators.required,Validators.pattern(this.regexes['number']['regex'])]);
  this.form.controls['fullname'].setValidators([Validators.required,Validators.pattern(this.regexes['fullname']['regex'])]);
  this.form.controls['callingname'].setValidators([Validators.required,Validators.pattern(this.regexes['callingname']['regex'])]);
  this.form.controls['gender'].setValidators([Validators.required]);
  this.form.controls['nic'].setValidators([Validators.required,Validators.pattern(this.regexes['nic']['regex'])]);
  this.form.controls['dob'].setValidators([Validators.required]);
  this.form.controls['photo'].setValidators([Validators.required]);
  this.form.controls['address'].setValidators([Validators.required,Validators.pattern(this.regexes['address']['regex'])]);
  this.form.controls['mobile'].setValidators([Validators.required,Validators.pattern(this.regexes['mobile']['regex'])]);
  this.form.controls['land'].setValidators([Validators.required,Validators.pattern(this.regexes['land']['regex'])]);
  this.form.controls['designation'].setValidators([Validators.required]);
  this.form.controls['assignmentdate'].setValidators([Validators.required]);
  this.form.controls['description'].setValidators([Validators.required,Validators.pattern(this.regexes['description']['regex'])]);
  this.form.controls['employeestatus'].setValidators([Validators.required]);

  Object.values(this.form.controls).forEach(control => { control.markAsTouched();});
}
```



```
initialize() {

  this.createView();

  this.gs.getAllList().then((gens: Gender[]) => {
    this.genders = gens;
    console.log("G-" + this.genders);
  });

  this.ds.getAllList().then((dess: Designation[]) => {
    this.designations = dess;
    console.log("D-" + this.designations);
  });

  this.ss.getAllList().then((stes: Employeestatus[]) => {
    this.employeestatuses = stes;
    console.log("S-" + this.employeestatuses);
  });

  this.rs.get('employee').then((regs: []) => {
    this.regexes = regs;
    console.log("R-" + this.regexes['number']['regex']);
    this.createForm();
  });

  this.createSearch();
}
```

Test for Errors → All Required fields will get Red color as their initial values set to empty string



Employee Detail

Number*

Full Name*

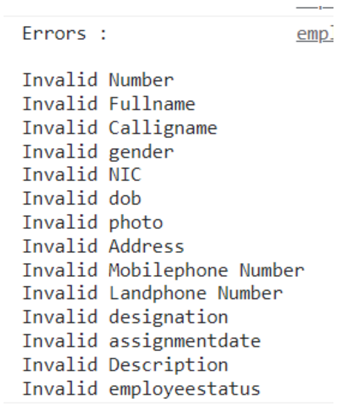
Calling Name*

Gender*

NIC*

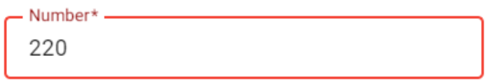
Birth Date*

Click add button without filling any form field



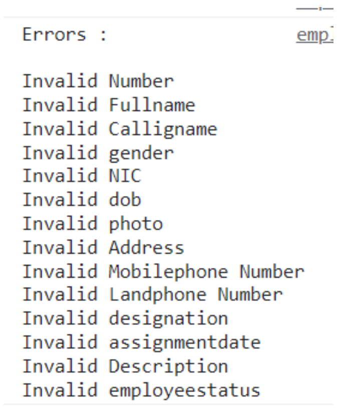
```
Errors :                     emp:

Invalid Number
Invalid Fullname
Invalid Callingname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

Fill one by one with valid values and detect Errors

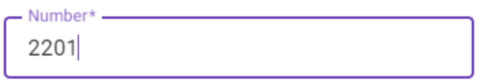


Number*

220

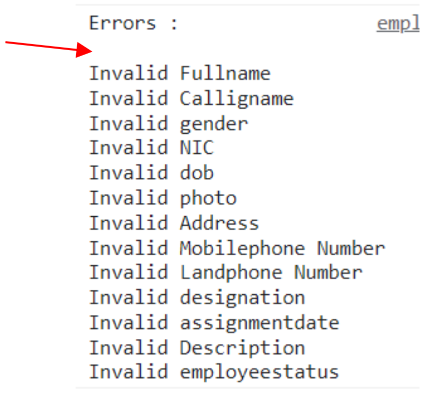


```
Errors :                     emp:

Invalid Number
Invalid Fullname
Invalid Callingname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```



Number*

2201



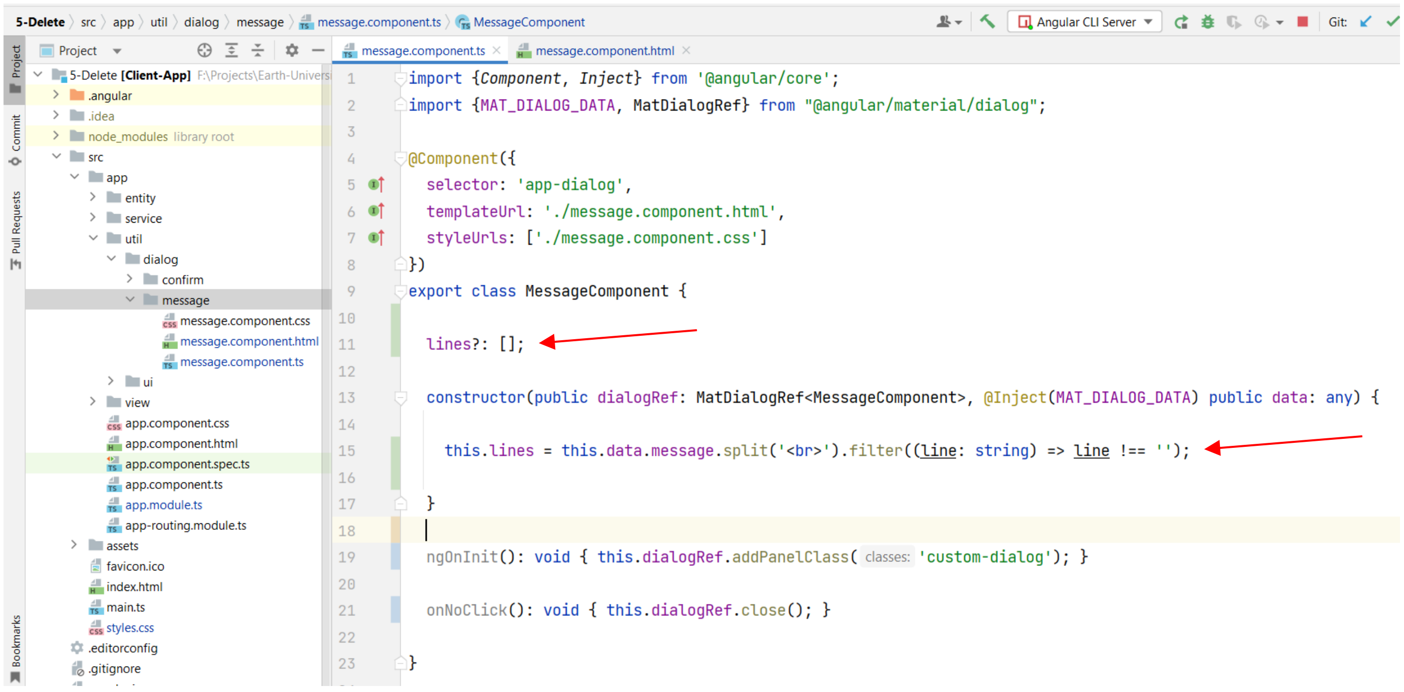
```
Errors :                     empl

Invalid Fullname
Invalid Callingname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus
```

Exercise-1 – **Change the Validation Colors in both Focus and Non Focus modes**
Use Global style.css File

Exercise-2 – **Re implement the above “createForm()” using a For Loop with Controls of the Form Group**
Follow the coding fashion in “getErrors()”

Show Error Message as Dialog Message

(1) Change the Message Component to Support Multiline Messages

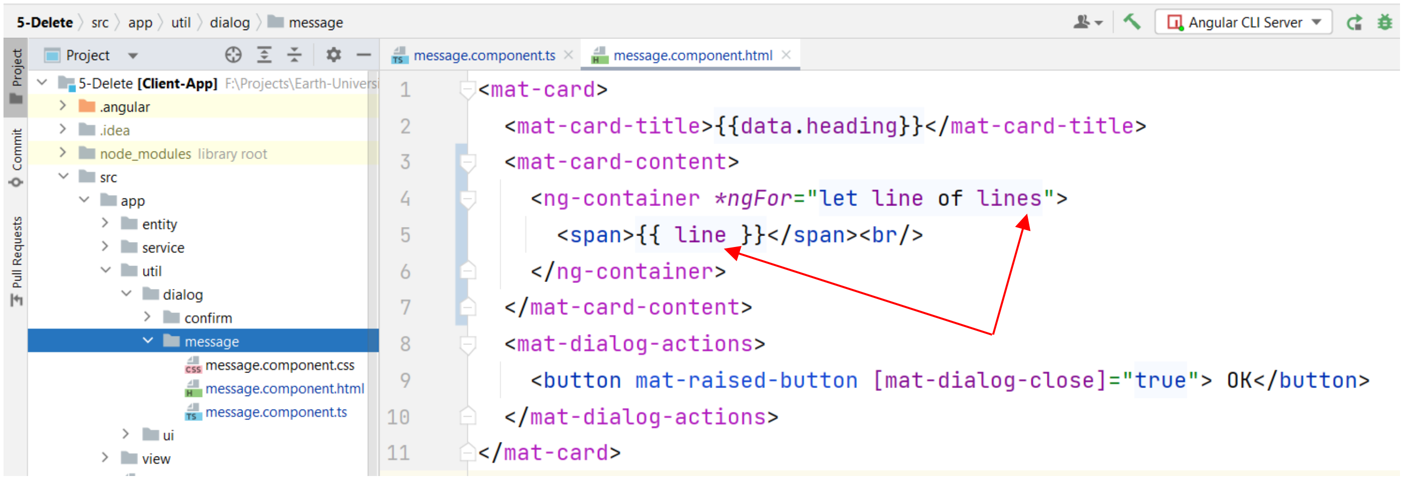


5-Delete 〉 src 〉 app 〉 util 〉 dialog 〉 message 〉 message.component.ts 〉 MessageComponent

```
5-Delete [Client-App] F:\Projects\Earth-Univers
  .angular
  .idea
  node_modules library root
  src
    app
      entity
      service
      util
        dialog
          confirm
          message
            message.component.css
            message.component.html
            message.component.ts
        ui
      view
      app.component.css
      app.component.html
      app.component.spec.ts
      app.component.ts
      app.module.ts
      app-routing.module.ts
    assets
    favicon.ico
    index.html
    main.ts
    styles.css
  .editorconfig
  .gitignore
```

message.component.ts

```
import {Component, Inject} from '@angular/core';
import {MAT_DIALOG_DATA, MatDialogRef} from "@angular/material/dialog";

@Component({
  selector: 'app-dialog',
  templateUrl: './message.component.html',
  styleUrls: ['./message.component.css']
})
export class MessageComponent {

  lines?: [];

  constructor(public dialogRef: MatDialogRef<MessageComponent>, @Inject(MAT_DIALOG_DATA) public data: any) {

    this.lines = this.data.message.split('<br>').filter((line: string) => line !== '');

  }

  ngOnInit(): void { this.dialogRef.addPanelClass( classes: 'custom-dialog'); }

  onNoClick(): void { this.dialogRef.close(); }

}
```



5-Delete 〉 src 〉 app 〉 util 〉 dialog 〉 message

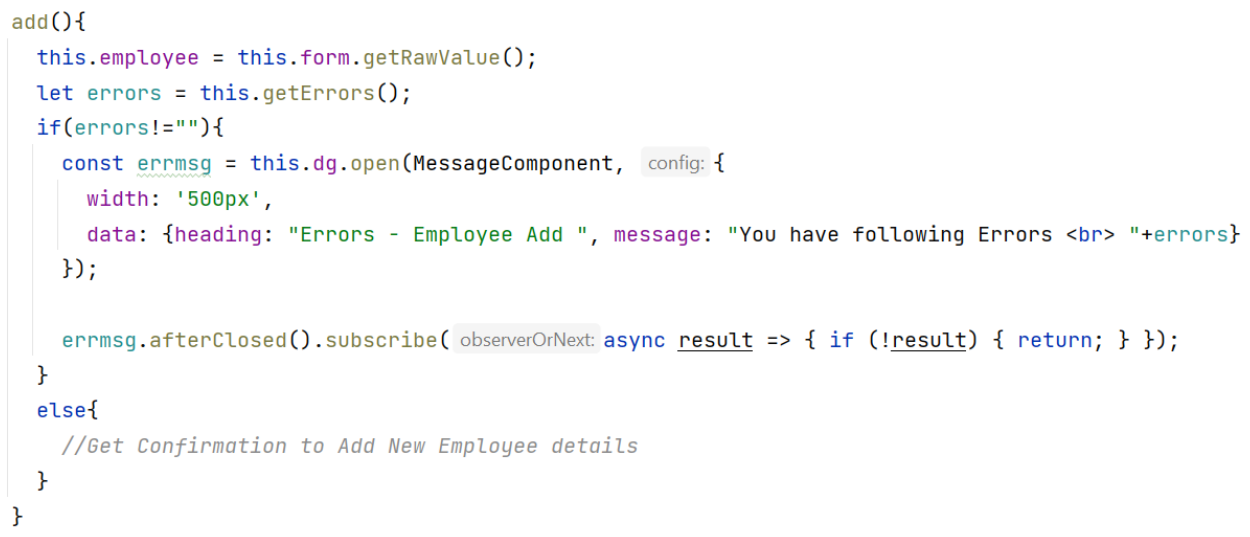
message.component.html

```
<mat-card>
  <mat-card-title>{{data.heading}}</mat-card-title>
  <mat-card-content>
    <ng-container *ngFor="let line of lines">
      <span>{{ line }}</span><br/>
    </ng-container>
  </mat-card-content>
  <mat-dialog-actions>
    <button mat-raised-button [mat-dialog-close]="true"> OK</button>
  </mat-dialog-actions>
</mat-card>
```

(2) Change the "getErrors()" method as follows

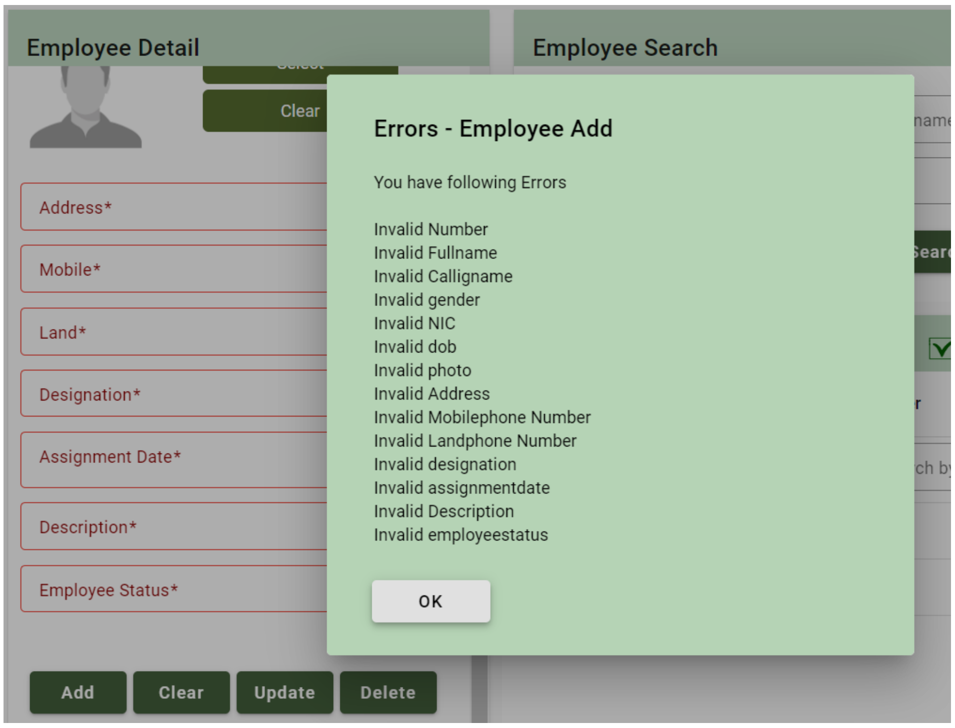
```
getErrors(): string {
  let errors: string="";
  for (const controlName in this.form.controls) {
    const control = this.form.controls[controlName];
    if (control.errors) {
      if(this.regexes[controlName]!=undefined)
      { errors=errors+"<br>"+ this.regexes[controlName]['message']; }
      else
      {errors=errors+"<br>Invalid "+ controlName;}
    }
  }
  return errors;
}
```

(3) Add Message to the “add()” method to display the Error Message



```
add(){
  this.employee = this.form.getRawValue();
  let errors = this.getErrors();
  if(errors!=""){
    const errmsg = this.dg.open(MessageComponent, config: {
      width: '500px',
      data: {heading: "Errors - Employee Add ", message: "You have following Errors <br> "+errors}
    });

    errmsg.afterClosed().subscribe( observerOrNext: async result => { if (!result) { return; } });
  }
  else{
    //Get Confirmation to Add New Employee details
  }
}
```

Testing



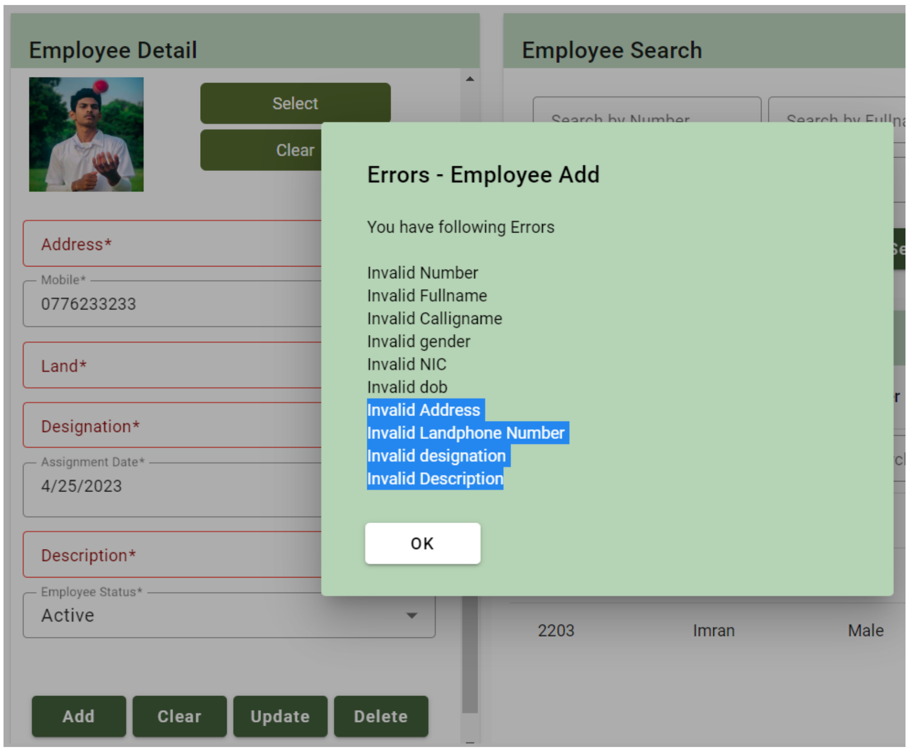
Errors - Employee Add

You have following Errors

Invalid Number
Invalid Fullname
Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid photo
Invalid Address
Invalid Mobilephone Number
Invalid Landphone Number
Invalid designation
Invalid assignmentdate
Invalid Description
Invalid employeestatus

OK

Reduce One by one by filing the each field with valid details

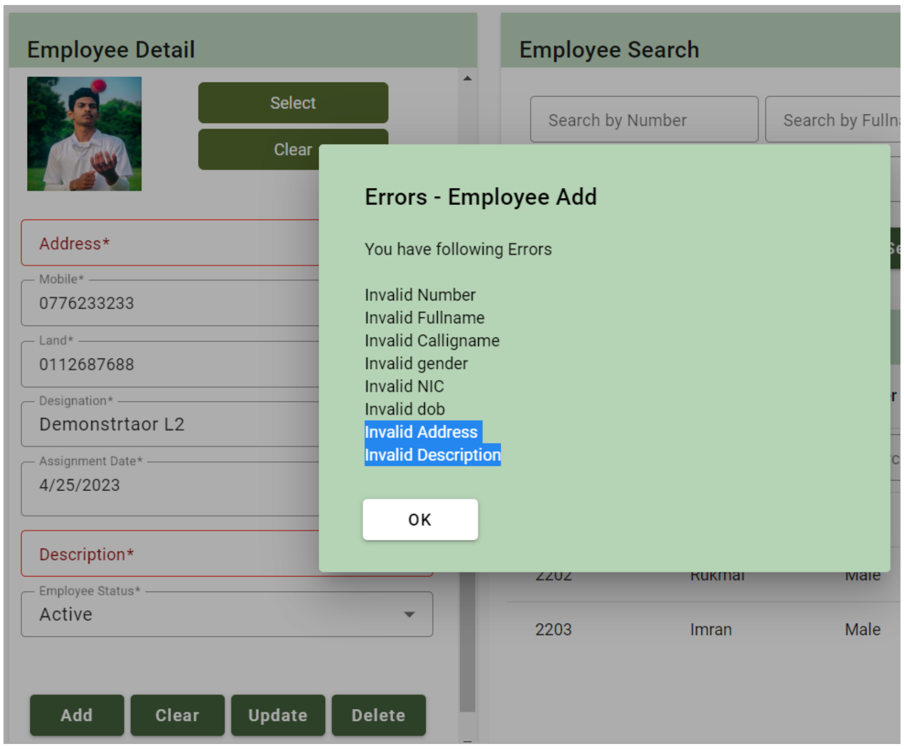


Errors - Employee Add

You have following Errors

Invalid Number
Invalid Fullname
Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid Address
Invalid Landphone Number
Invalid designation
Invalid Description

OK



Errors - Employee Add

You have following Errors

Invalid Number
Invalid Fullname
Invalid Calligname
Invalid gender
Invalid NIC
Invalid dob
Invalid Address
Invalid Description

OK