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

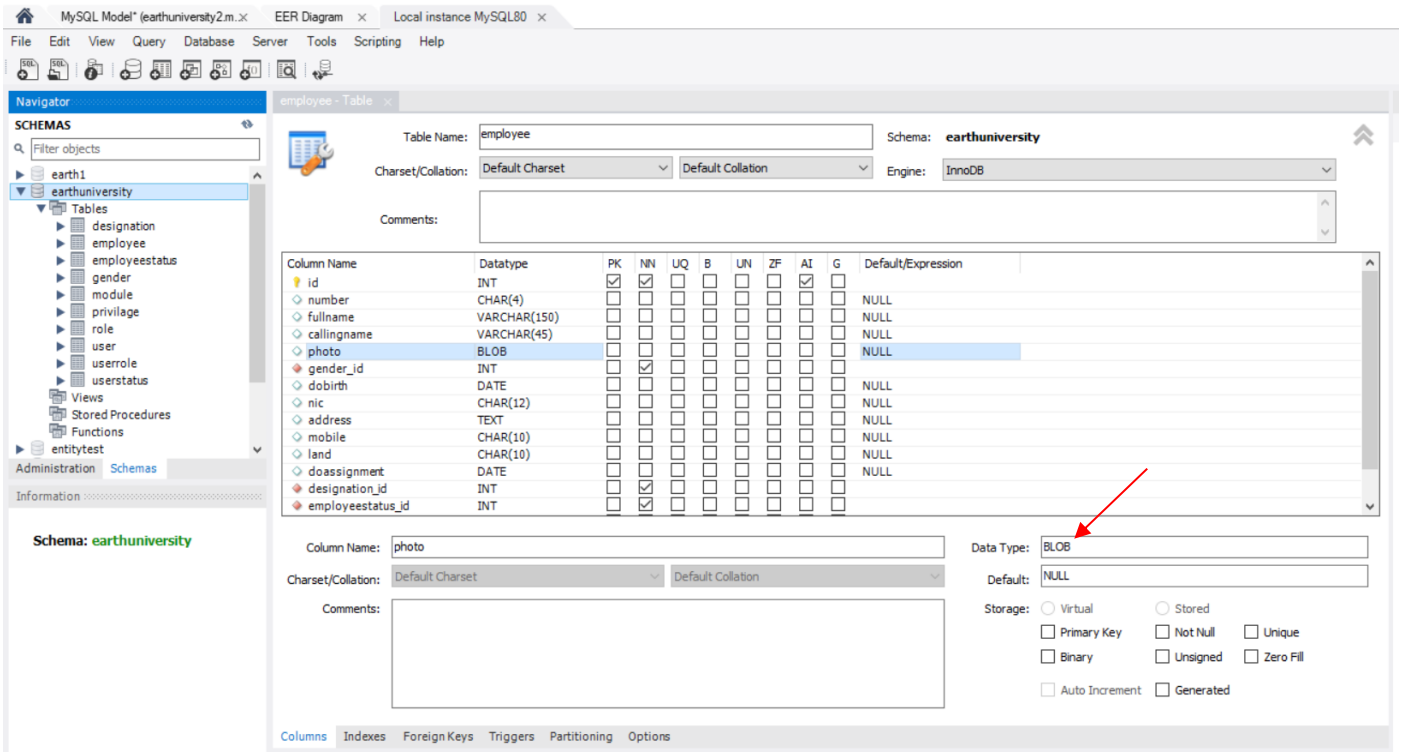
==========

1. **Update Database → Change Data Type of the "photo" from BLOB to LONGBLOB**
2. **Test for Confirmation Message**
3. **Service Implementation with POST Request**
4. **Show Error Message while typing**

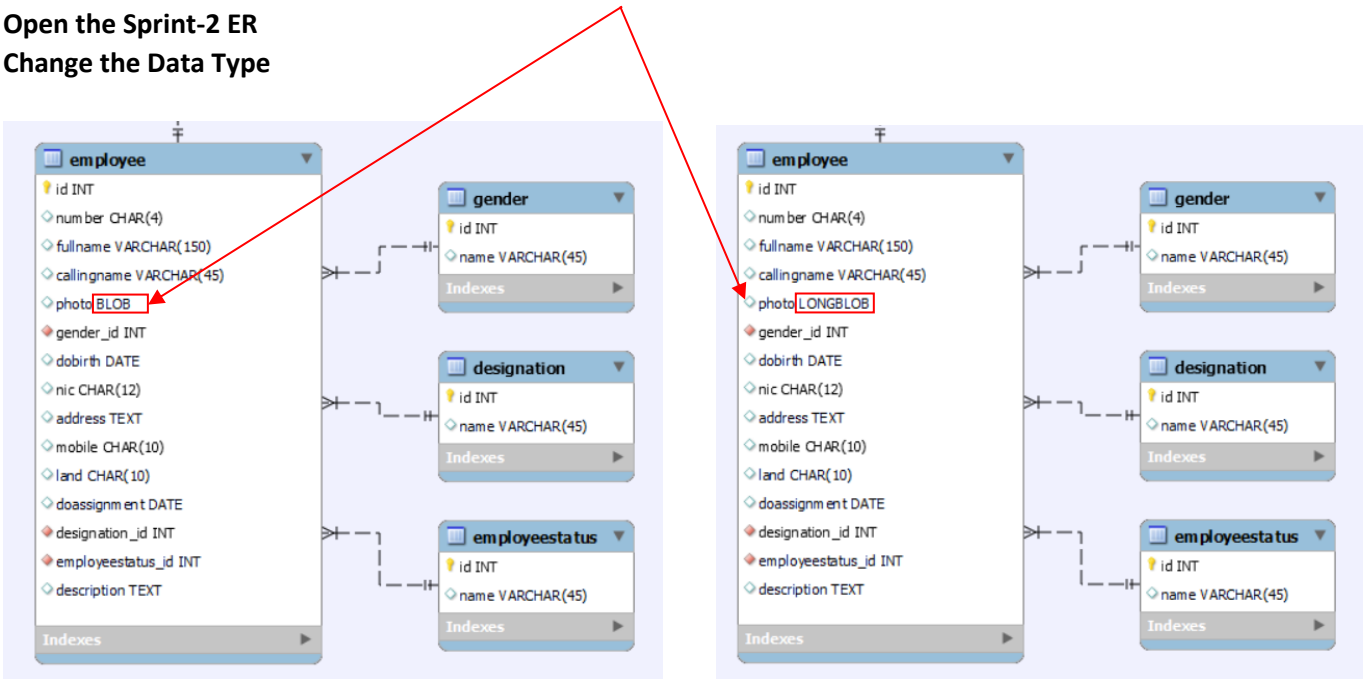
1. Update Database → Change Data Type of the “photo” from BLOB to LONGBLOB

Change the Data Type of the “photo” column from “BLOB” to “LONGBLOB”

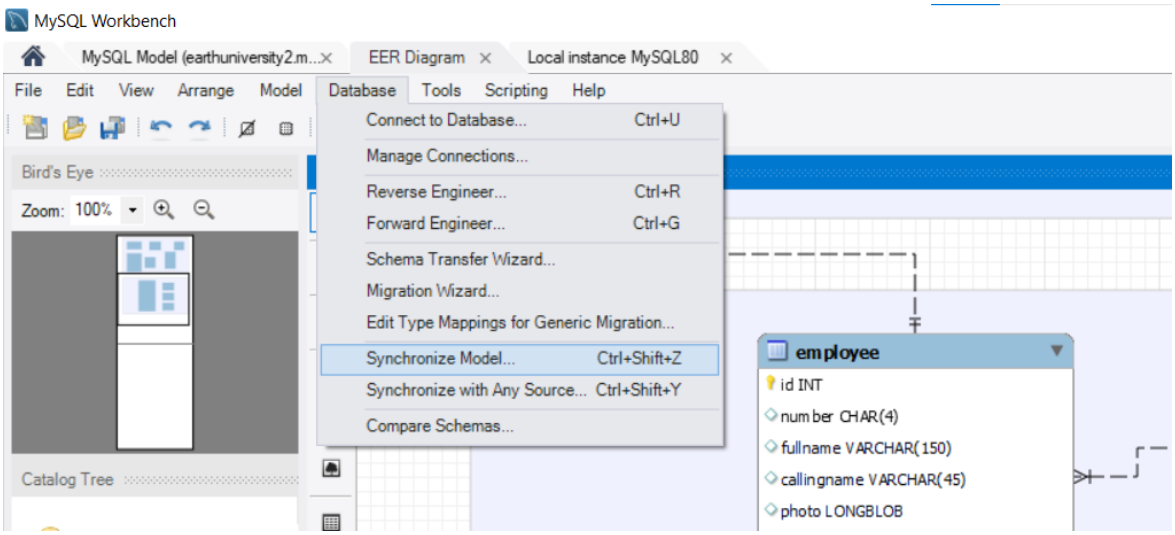
You can change it using Admin Panel of the MySQL Work Bench
(But this is not recommended as the ER-Model and Database get in consisted otherwise)

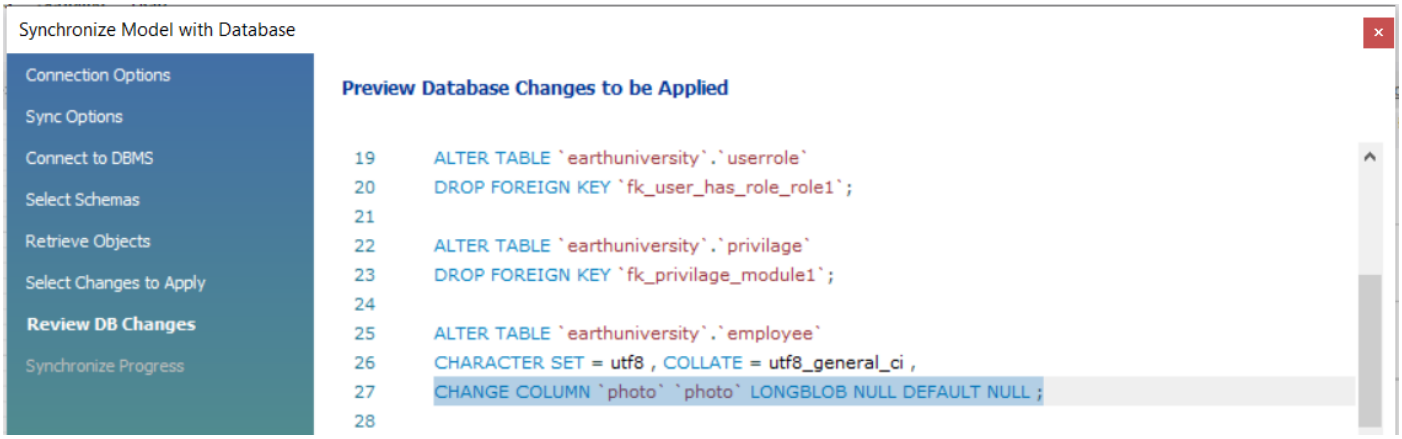


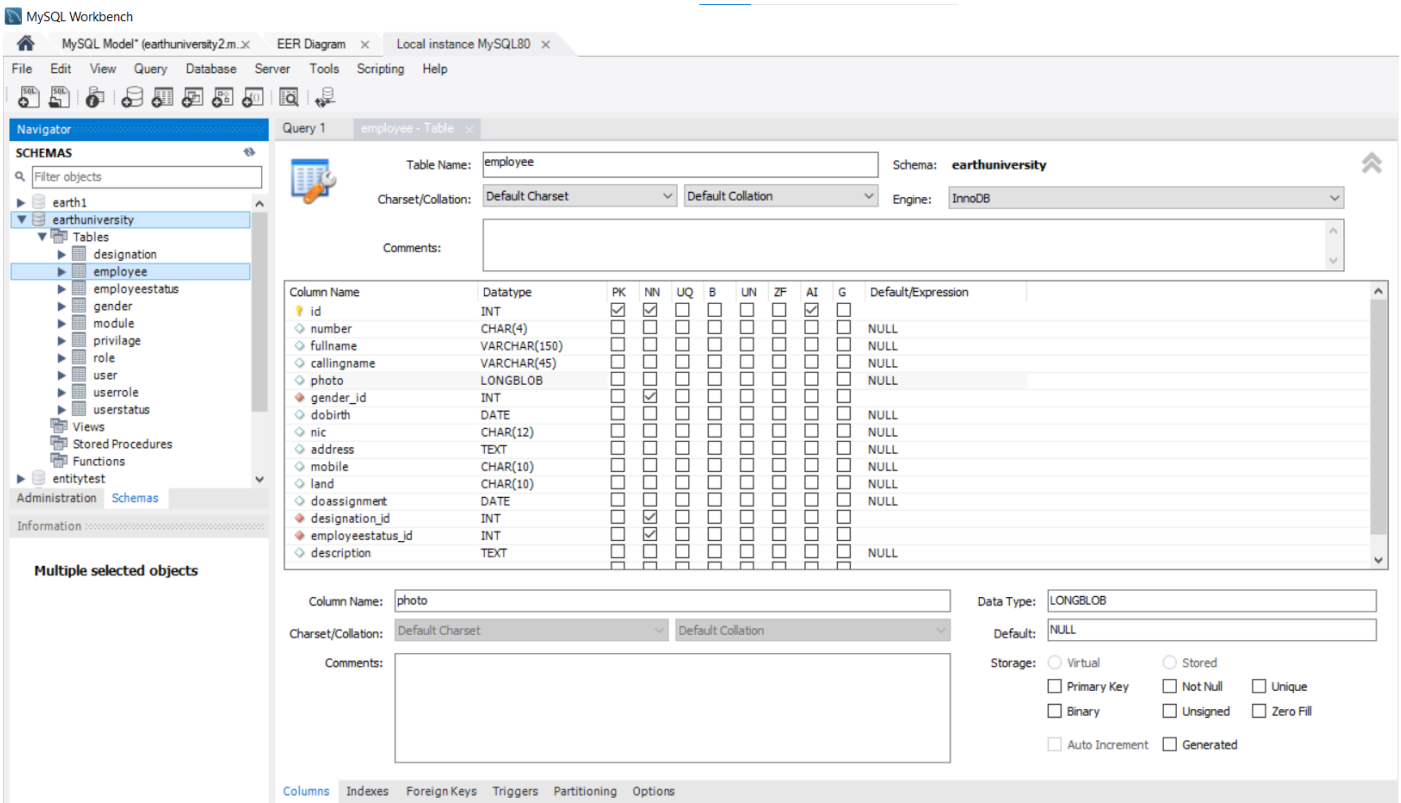
Open the Sprint-2 ER
Change the Data Type



Save Changes → Synchronize Model [Database Tab → Synchronize Model]







This is important as the Size of the selected photo may exceeds the capacity of BLOB otherwise.

HTML Error Correction

```
<mat-form-field appearance="outline">
  <mat-label>Gender</mat-label>
  <mat-select formControlName="gender">
    <mat-option [value]="null" selected>Not Selected</mat-option>
    <mat-option *ngFor="let gender of genders" [value]="gender">{{gender.name}}</mat-option>
  </mat-select>
</mat-form-field>
```

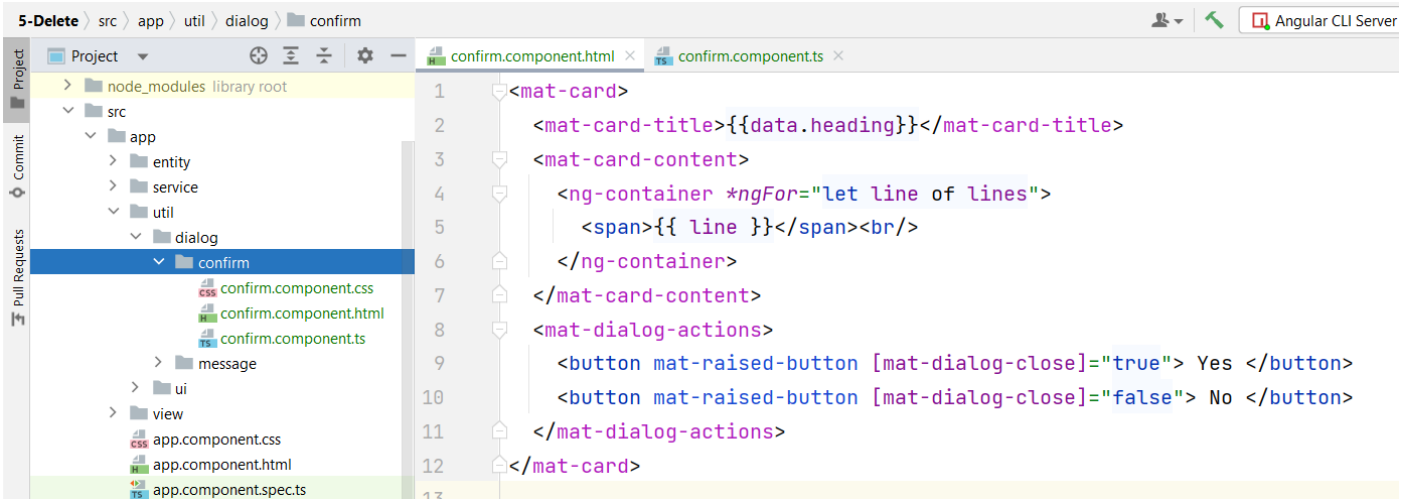
```
<mat-form-field appearance="outline">
  <mat-label>Designation</mat-label>
  <mat-select formControlName="designation">
    <mat-option [value]="null" selected>Not Selected</mat-option>
    <mat-option *ngFor="let designation of designations" [value]="designation">{{designation.name}}</mat-option>
  </mat-select>
</mat-form-field>
```

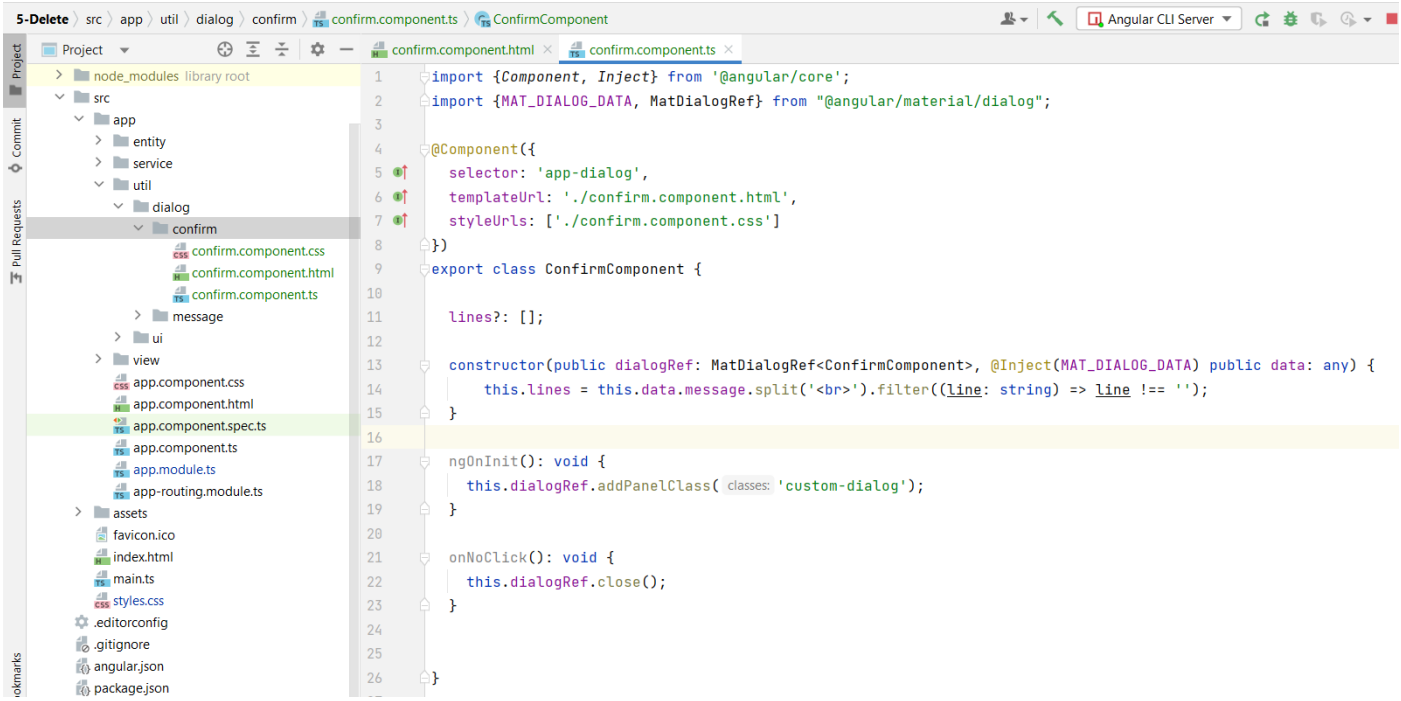
```
<mat-form-field appearance="outline">
  <mat-label>Employee Status</mat-label>
  <mat-select formControlName="employeestatus">
    <mat-option [value]="null" selected>Not Selected</mat-option>
    <mat-option *ngFor="let employeestatus of employeestatuses" [value]="employeestatus">{{employeestatus.name}}</mat-option>
  </mat-select>
</mat-form-field>
```

1. Update Database → Change Data Type of the "photo" from BLOB to LONGBLOB
2. **Test for Confirmation Message**
3. **Service Implementation with POST Request**
4. **Show Error Message while typing**

2. Test for Confirmation Message

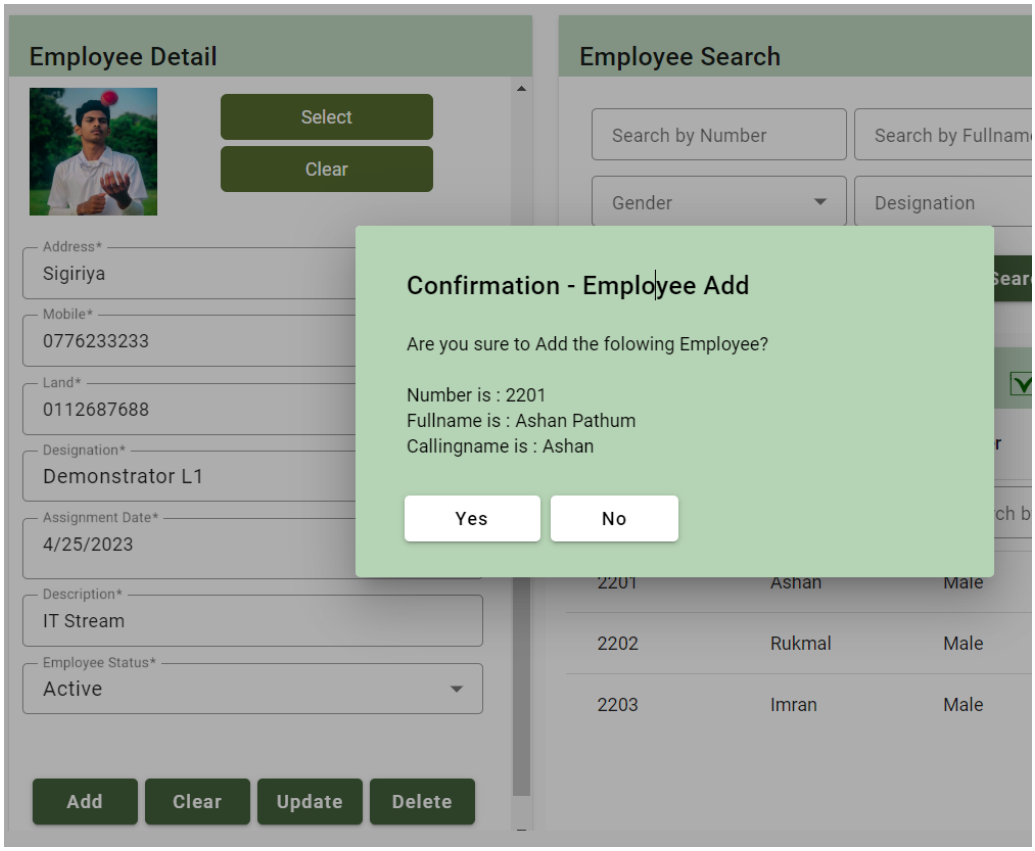
1. **Update Confirmation Component to show multi Line Description as same as previous Module**





1. Update “add()” method to show the confirmation message with employee details to be added

```
add(){

  let errors = this.getErrors();

  if(errors!=""){

    const errmsg = this.dg.open(MessageComponent, {
      width: '500px',
      data: {heading: "Errors - Employee Add ", message: "You have following Errors <br>
"+errors}
    });

    errmsg.afterClosed().subscribe(async result => { if (!result) { return; } });
  }
  else{

    this.employee = this.form.getRawValue();
    //console.log("Photo-Before"+this.employee.photo);
    this.employee.photo=btoa(this.imageempurl);
    //console.log("Photo-After"+this.employee.photo);

    let empdata: string = "";

    empdata = empdata + "<br>Number is : "+ this.employee.number;
    empdata = empdata + "<br>Fullname is : "+ this.employee.fullname;
    empdata = empdata + "<br>Callingname is : "+ this.employee.callingname;

    const confirm = this.dg.open(ConfirmComponent, {
      width: '500px',
      data: {heading: "Confirmation - Employee Add", message: "Are you sure to Add the folowing
Employee? <br> <br>"+ empdata}
    });

    confirm.afterClosed().subscribe(async result => {
      if(result){
        console.log("EmployeeService.add(emp)");
        this.form.reset();
        this.clearImage();
      }
    });
  }
}
```



Test Cases **No → Form data will be intact with the form**
Yes → Form data will be cleared and Print Console Output ("EmployeeService.add(emp)");

1. Update Database → Change Data Type of the "photo" from BLOB to LONGBLOB
2. Test for Confirmation Message
3. **Service Implementation with POST Request**
4. **Show Error Message while typing**

3. Service Implementation with POST Request

(a) Implement add function in EmployeeService

```
export class EmployeeService {

  constructor(private http: HttpClient) {  }

  async getAll(query:string): Promise<Array<Employee>> {
    const employees = await this.http.get<Array<Employee>>( url: 'http://localhost:8080/employees'+query).toPromise();
    if(employees == undefined){
      return [];
    }
    return employees;
  }

  async add(employee: Employee): Promise<[]|undefined>{
    //console.log("Employee Adding-"+JSON.stringify(employee));
    return this.http.post<[]>( url: 'http://localhost:8080/employees', employee).toPromise();
  }


}
```

(b) Call it in add function in EmployeeComponnet

```
add(){

  let errors = this.getErrors();

  if(errors!=""){

    const errmsg = this.dg.open(MessageComponent, {
      width: '500px',
      data: {heading: "Errors - Employee Add ", message: "You have following Errors <br> "+errors}
    });

    errmsg.afterClosed().subscribe(async result => { if (!result) { return; } });
  }
  else{

    this.employee = this.form.getRawValue();
    //console.log("Photo-Before"+this.employee.photo);
    this.employee.photo=btoa(this.imageempurl);
    //console.log("Photo-After"+this.employee.photo);

    let empdata: string = "";

    empdata = empdata + "<br>Number is : "+ this.employee.number;
    empdata = empdata + "<br>Fullname is : "+ this.employee.fullname;
    empdata = empdata + "<br>Callingname is : "+ this.employee.callingname;

    const confirm = this.dg.open(ConfirmComponent, {
      width: '500px',
      data: {heading: "Confirmation - Employee Add",
             message: "Are you sure to Add the folowing Employee? <br> <br>"+ empdata}
    });

    let addstatus:boolean=false;
    let addmessage:string="Server Not Found";

    confirm.afterClosed().subscribe(async result => {
      if(result){
       // console.log("EmployeeService.add(emp)");

        this.es.add(this.employee).then((responce: []|undefined) => {
          console.log("Res-"+responce);
          console.log("Un-"+responce==undefined);
          if(responce!=undefined){ // @ts-ignore
           console.log("Add-"+responce['id']+"-"+responce['url']+"-"+(responce['errors']==""));
            // @ts-ignore
           addstatus = responce['errors']=="";
            console.log("Add Sta-"+addstatus);
            if(!addstatus) { // @ts-ignore
              addmessage=responce['errors'];
            }
          }
          else{
            console.log("undefined");
            addstatus=false;
            addmessage="Content Not Found"
          }
        }).finally( () =>{

            if(addstatus) {
              addmessage = "Successfully Saved";
              this.form.reset();
              this.clearImage();
            }

            const stsmsg = this.dg.open(MessageComponent, {
              width: '500px',
              data: {heading: "Status -Employee Add", message: addmessage}
            });

            stsmsg.afterClosed().subscribe(async result => { if (!result) { return;} }) ;} );
      }
    });
  }
}
```

Most of the codes in the above will be integrated into the UiAssist

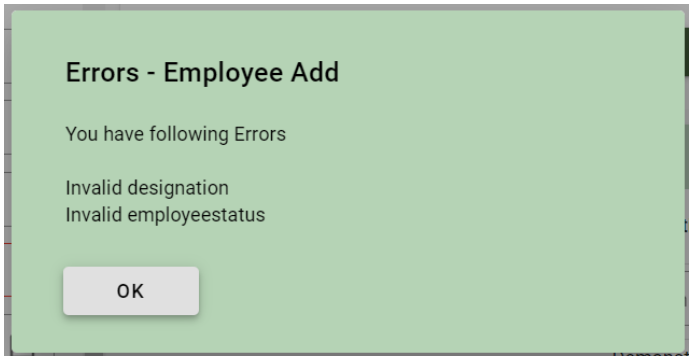
But it is very important to explain the behavior of these codes for explanation and modification stages

Test Case – 1 Client Form Validation Errors
Test Case – 2 Client Confirmation Message

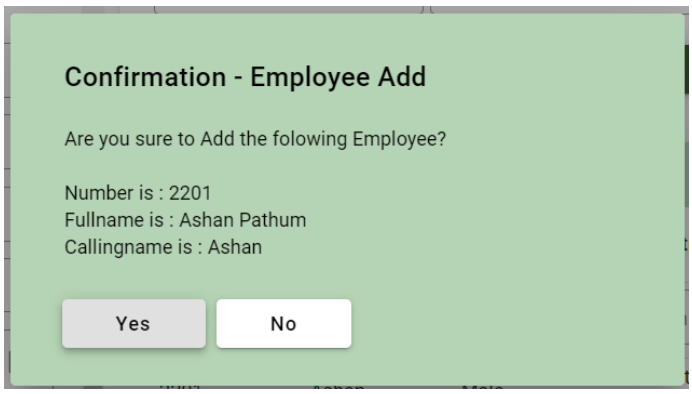
You can't test the following Test Cases until Server-App completed in next Document

Test Case – 3 Successfully Saved
Test Case – 4 Content Not Available (May/May Not Successfully Saved)
Test Case – 5 Server Not Available
Test Case – 6 Service Not Available
Test Case – 7 Server Validation Failed

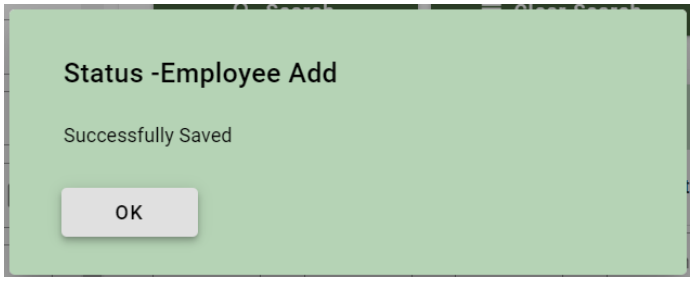
Test Case – 1 Client Form Validation Errors



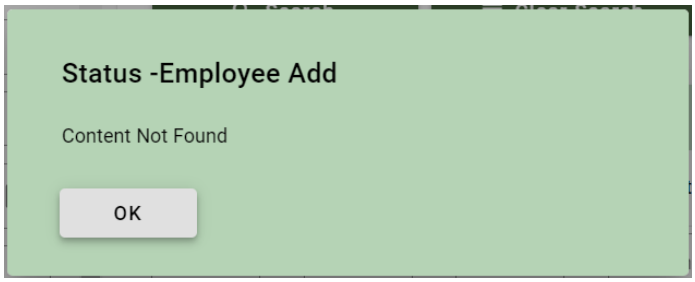
Test Case – 2 Client Confirmation Message



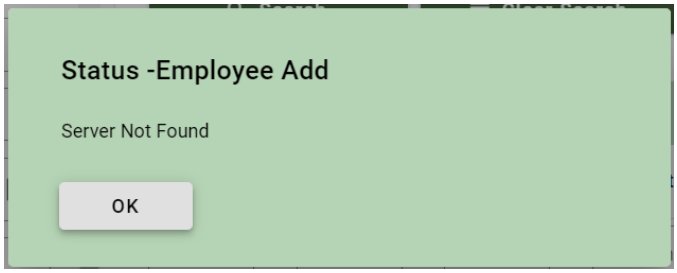
Test Case – 3 Successfully Saved



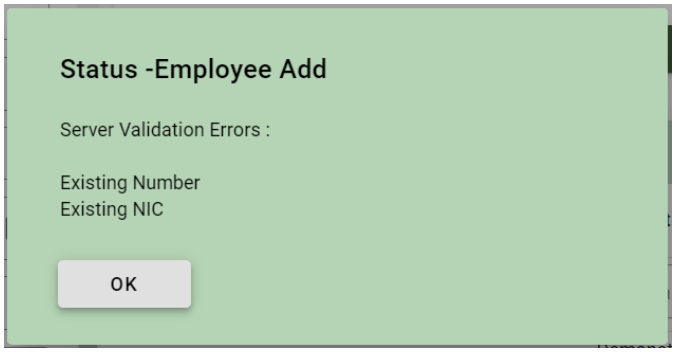
Test Case – 4 Content Not Available (May/May Not Successfully Saved)



Test Case – 5 Server Not Available



Test Case – 7 Server Validation Failed



1. Update Database → Change Data Type of the "photo" from BLOB to LONGBLOB
2. Test for Confirmation Message
3. Service Implementation with POST Request
4. **Show Error Message while typing**

4. Show Error Message while typing

However we do not use this type of error indicating as it uses much space n between control fields
You need to adjust the gap between input fields to show the error clearly with proper spacing
Use CSS for the purpose if you need.
We use <p><p> temporally for the same purpose.
Hear we use the same Error Messages received with regexes array
(But in this case we must load the regexes synchronously in the constructor to remove all possible errors)

```
<mat-form-field appearance="outline">
  <mat-label>Number</mat-label>
  <input matInput formControlName="number">
  <mat-error *ngIf="this.form.controls['number'].invalid"> {{ this.regexes['number']['message'] }}</mat-error>
</mat-form-field>
  <p></p>
```

1. Update Database → Change Data Type of the "photo" from BLOB to LONGBLOB
2. Test for Confirmation Message
3. Service Implementation with POST Request
4. Show Error Message while typing

==========

2(d) Employee Insert

6. Folder and Project preparation with Backups
7. Prepare Supportive Data for the Insert Form
8. Prepare Regex(Regular Expression) Service for the Insert Form

9. Client App
    6. Define Form Controls with Required Validator
    7. Define HTML Form Controls with relevant Binding
    8. Test for Error Messages
    9. Test for Confirmation Message
    10. Service Implementation with POST Request

10. **Server-App with Server Error Messages**

==========

Department of Information Technology
Earth University College