FORM AND SERVER SYSTEM

COURSE PROJECT REPORT

Submitted by

Dhruv Sahu (RA2011050010033)

Under the guidance of

Dr. M. PRAKASH

In partial fulfilment for the Course

of

18CSP102L –INDUSTRIAL TRAINING

in

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University u/s 3 of UGC Act, 1956)

KATTANKULATHUR - 603 203

November, 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this mini project titled "**Form and server system"** is the bonafide work of **Dhruv Sahu (RA2011050010033),** who carried out the project work under my supervision.

SUPERVISOR	**HEAD OF THE DEPARTMENT**
Dr. M. PRAKASH	Dr. M. LAKSHMI
Associate Professor	Professor & Head
Department of Data Science and Business Systems	Department of Data Science and Business Systems
SRM Institute of Science and Technology Kattankulathur – 603 203	SRM Institute of Science and Technology Kattankulathur – 603 203

ABSTRACT

This research work generally summarizes the activities carried out in the design and implementation of the form project with server system. It was carried out to train and debug angular apps that are specialized in routing and making changes in systems at the same time storing and updating values in database. In this project an employee can create, read, update his/her data and update it to the servers of the company.

The project’s designwas then constructed to visualize the attainment of these aims using Unified Modeling Language diagrams, Entity Relationship diagram, as well as Data Flow Diagram. In the implementation part of this project, the methodology was used for the development of this project. Also, visual studio Professional 2015 was used as an Integrated Development Environment (IDE), developed using HTML, CSS, Bootstrap, Angular, Node.js.

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professor Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to my Course project Faculty Dr. **M. Prakash, Associate Professor, Department of Data Science and Business Systems,** for his assistance, timely suggestion and guidance throughout the duration of this course project.

We extend our gratitude to our **HoD, Dr. M. Lakshmi, Professor, Department of Data Science and Business Systems,** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

TABLE OF CONTENTS

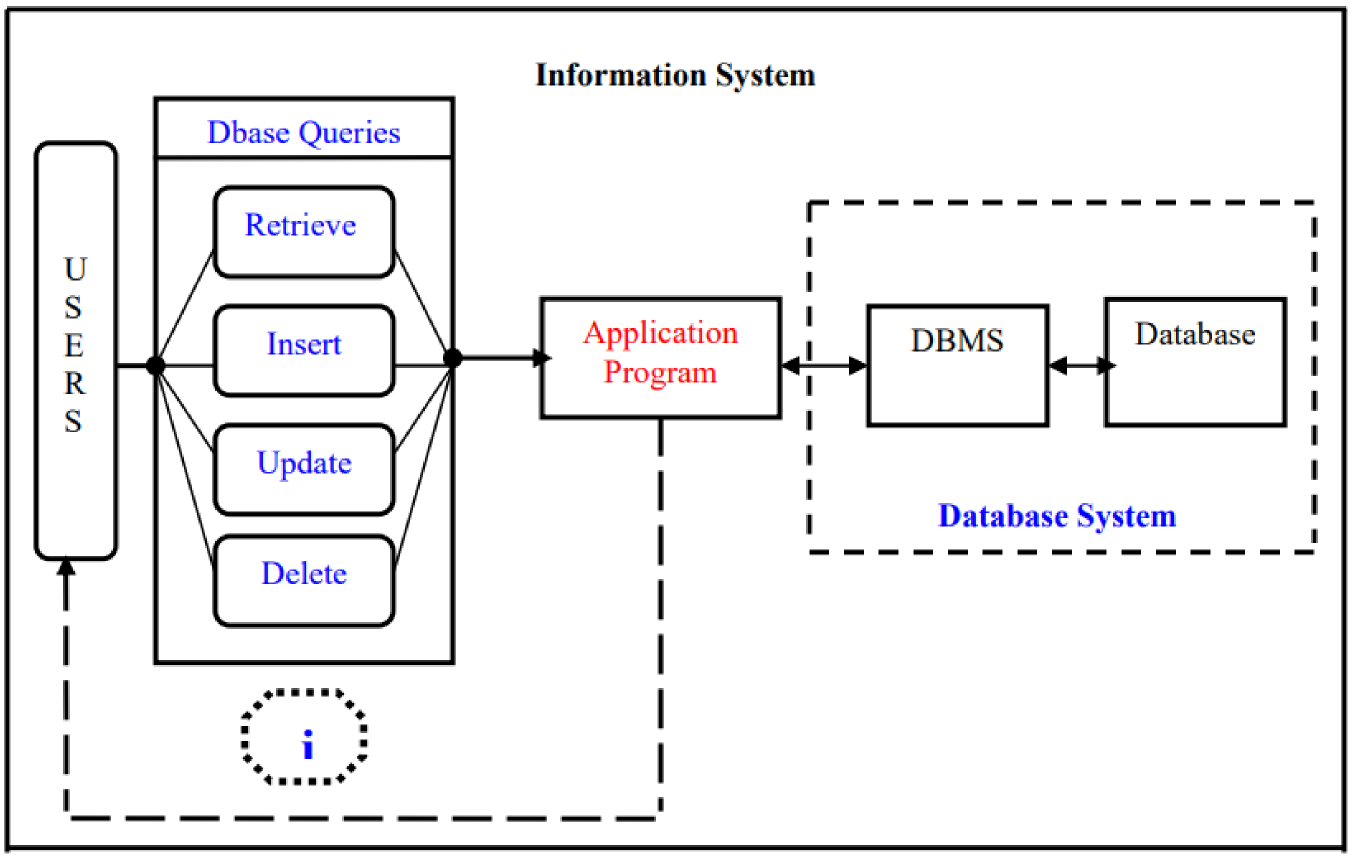
INTRODUCTION

Employee management system is an application based system, having two applications developed, one for employers to manage employee details and another for employees to mark their attendance. Every organization whether government or private uses an information system to store data of their staff. However, in India it is found that many small scale industries use pen and paper to keep a record. However, there are many advanced technology systems available that can do this work but they all are costly for these low level industries. Employee Management system is an application that enables users to create and store Employee Records. The application also provides facilities of a payroll system which enables user to generate Pay slips too. This application is helpful to department of the organization which maintains data of employees related to an organization . In this world of growing technologies everything has been computerized. With large number of work opportunities the Human workforce has increased. Thus there is a need of a system which can handle the data of such a large number of Employees in an organization. This project simplifies the task of maintain records because of its user friendly nature.

LITERATURE REVIEW

Most of the contemporary Information systems are based on the Database technology as a collection of logically related data, and DBMS as a software system allowing the users to define, create, maintain and control access to the database. The process of constructing such kind of systems is not so simple. It involves a mutual development of application program and database. The application program is actually the bridge between the users and the database, where the data is stored. Thus, the well-developed application program and database are very important for the reliability, flexibility and functionality of the system. The so defined systems differentiate to each other and their development comprises a great variety of tasks.

Architectural diagram:-



SYSTEM DESIGN AND IMPLEMENTATION

- Style.css
- Index.html

The above come under Angular components.

HTML:

The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.
HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.
HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.[2] A form of HTML, known as HTML5, is used to display video and audio, primarily using the <canvas> element, in collaboration with JavaScript.

CSS:-

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages thatshare the file and its formatting.

Separation of formatting and content also makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech- based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318. The W3C operates a free CSS validation service for CSS documents.

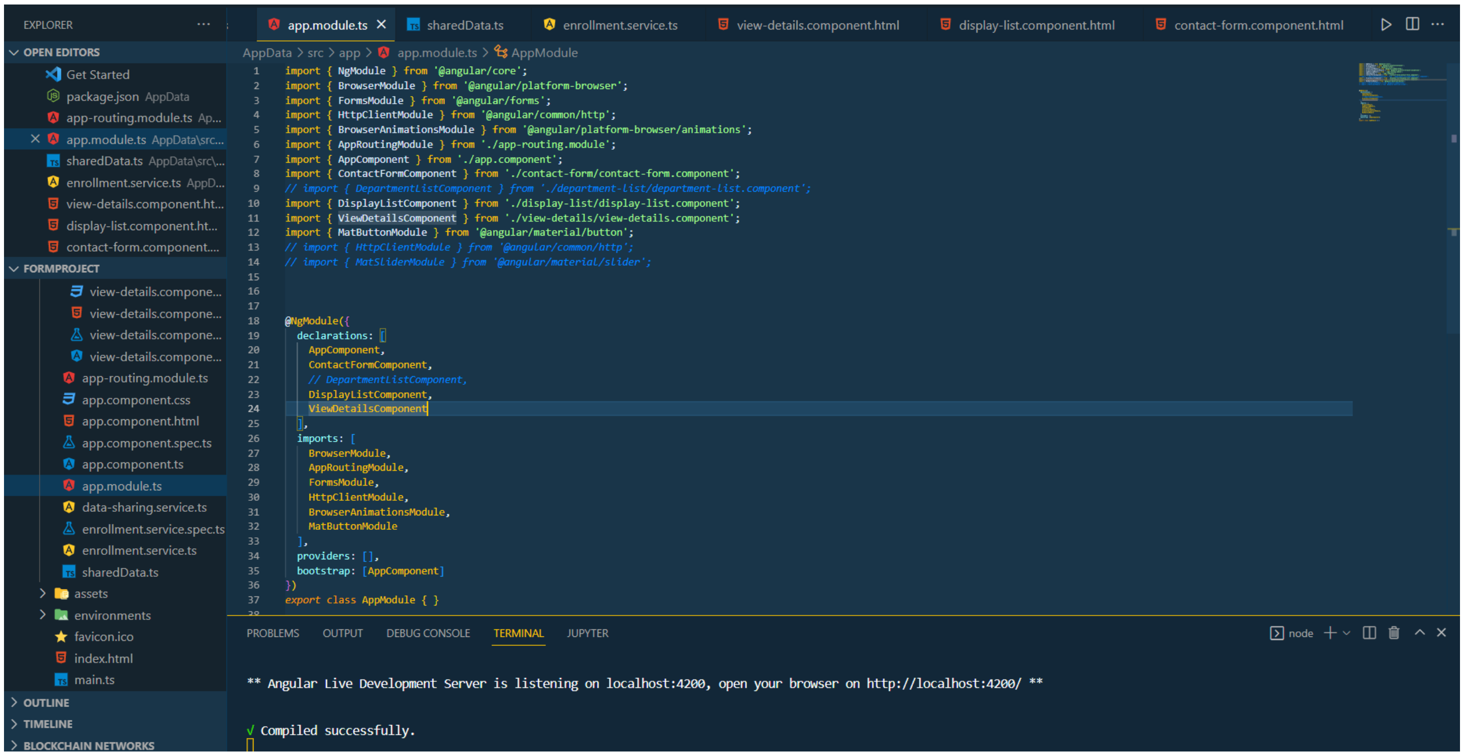
Angular:

Angular (also referred to as "Angular 2+") is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS.

Node.js:

Node.js is a cross-platform runtime environment and library for running JavaScript applications outside the browser. It is used for creating server-side and networking web applications. It is open source and free to use. It can be downloaded from this link https://nodejs.org/en/. Many of the basic modules of Node.js are written in JavaScript. Node.js is mostly used to run real-time server applications.The definition given by its official documentation is as follows: Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

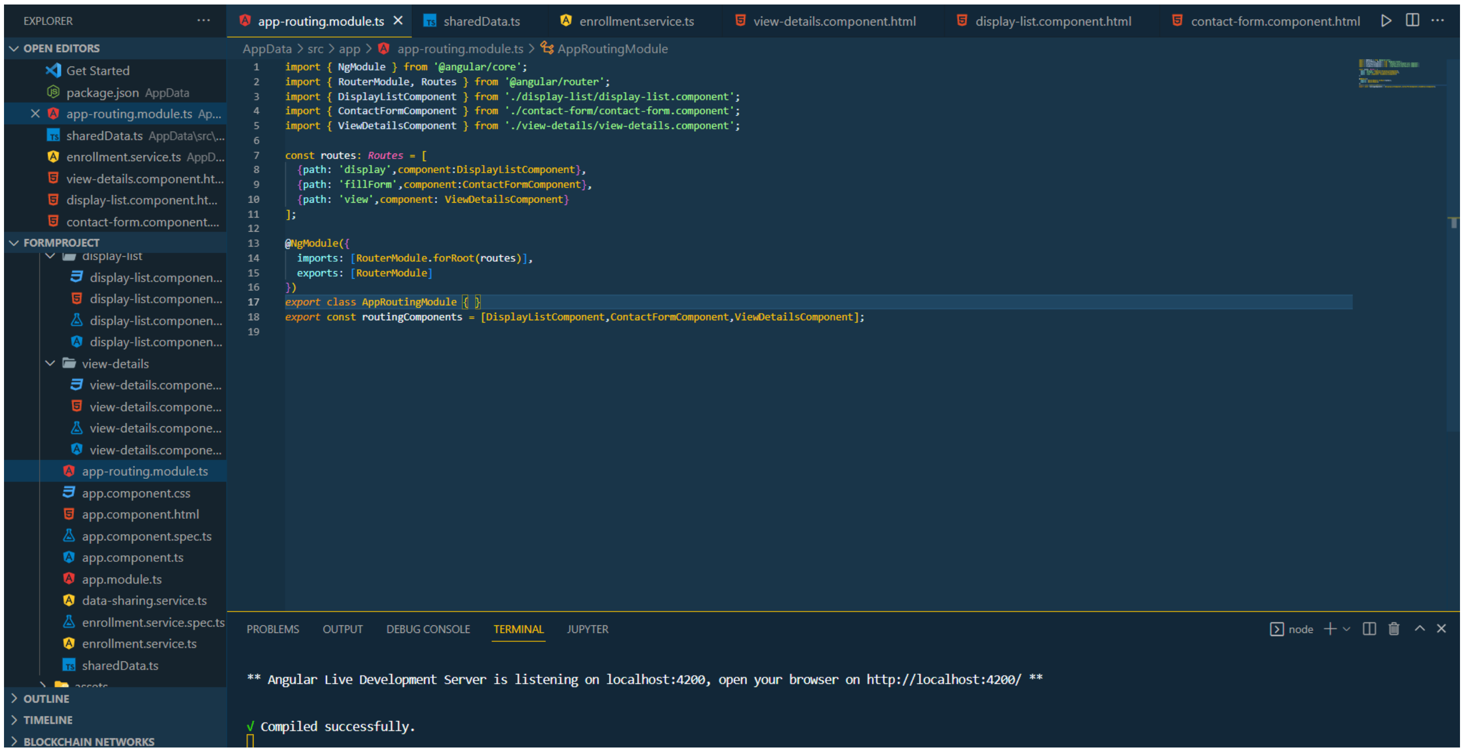
SOURCE CODE:-



AppData > src > app > app.module.ts > AppModule

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ContactFormComponent } from './contact-form/contact-form.component';
// import { DepartmentListComponent } from './department-list/department-list.component';
import { DisplayListComponent } from './display-list/display-list.component';
import { ViewDetailsComponent } from './view-details/view-details.component';
import { MatButtonModule } from '@angular/material/button';
// import { HttpClientModule } from '@angular/common/http';
// import { MatSliderModule } from '@angular/material/slider';



@NgModule({
  declarations: [
    AppComponent,
    ContactFormComponent,
    // DepartmentListComponent,
    DisplayListComponent,
    ViewDetailsComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
    BrowserAnimationsModule,
    MatButtonModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

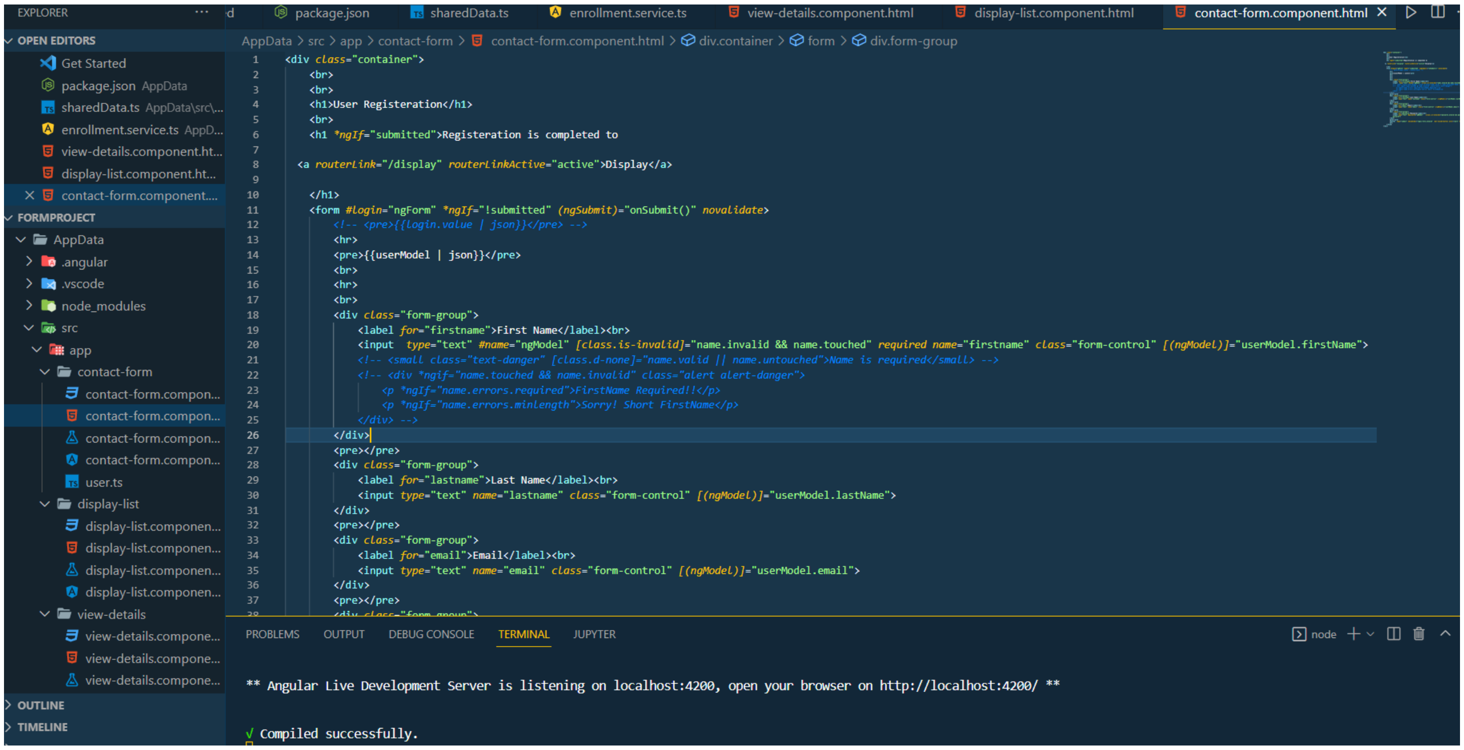
```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

√ Compiled successfully.
```



AppData > src > app > app-routing.module.ts > AppRoutingModule

```typescript
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { DisplayListComponent } from './display-list/display-list.component';
import { ContactFormComponent } from './contact-form/contact-form.component';
import { ViewDetailsComponent } from './view-details/view-details.component';

const routes: Routes = [
  {path: 'display',component:DisplayListComponent},
  {path: 'fillForm',component:ContactFormComponent},
  {path: 'view',component: ViewDetailsComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
export const routingComponents = [DisplayListComponent,ContactFormComponent,ViewDetailsComponent];

```

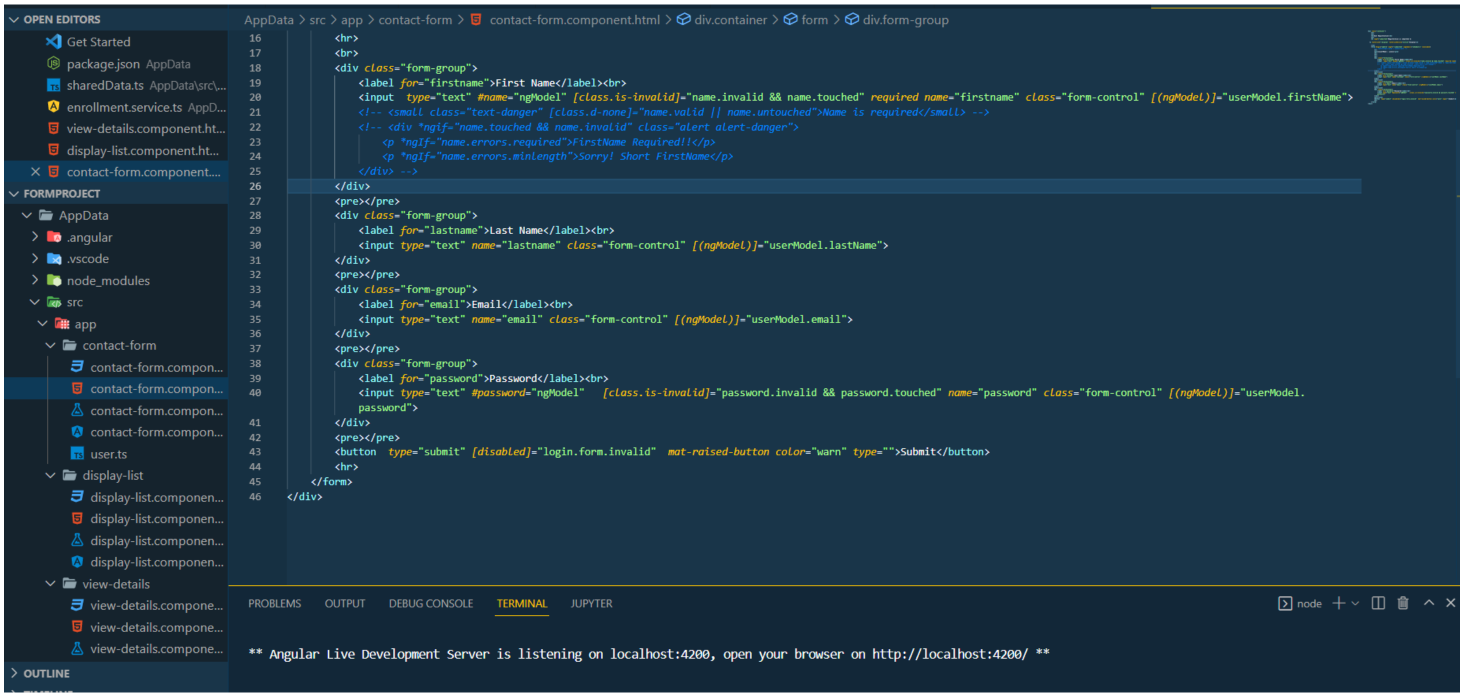
```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

√ Compiled successfully.
```


EXPLORER
package.json
sharedData.ts
enrollment.service.ts
view-details.component.html
display-list.component.html
contact-form.component.html
OPEN EDITORS
FORMPROJECT
OUTLINE
TIMELINE
AppData > src > app > contact-form > contact-form.component.html > div.container > form > div.form-group

```html
<div class="container">
    <br>
    <br>
    <h1>User Registeration</h1>
    <br>
    <h1 *ngIf="submitted">Registeration is completed to

  <a routerLink="/display" routerLinkActive="active">Display</a>

    </h1>
    <form #login="ngForm" *ngIf="!submitted" (ngSubmit)="onSubmit()" novalidate>
        <!-- <pre>{{login.value | json}}</pre> -->
        <hr>
        <pre>{{userModel | json}}</pre>
        <br>
        <hr>
        <br>
        <div class="form-group">
            <label for="firstname">First Name</label><br>
            <input  type="text" #name="ngModel" [class.is-invalid]="name.invalid && name.touched" required name="firstname" class="form-control" [(ngModel)]="userModel.firstName">
            <!-- <small class="text-danger" [class.d-none]="name.valid || name.untouched">Name is required</small> -->
            <!-- <div *ngif="name.touched && name.invalid" class="alert alert-danger">
                <p *ngIf="name.errors.required">FirstName Required!!</p>
                <p *ngIf="name.errors.minlength">Sorry! Short FirstName</p>
            </div> -->
        </div>
        <pre></pre>
        <div class="form-group">
            <label for="lastname">Last Name</label><br>
            <input type="text" name="lastname" class="form-control" [(ngModel)]="userModel.lastName">
        </div>
        <pre></pre>
        <div class="form-group">
            <label for="email">Email</label><br>
            <input type="text" name="email" class="form-control" [(ngModel)]="userModel.email">
        </div>
        <pre></pre>
```

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
node

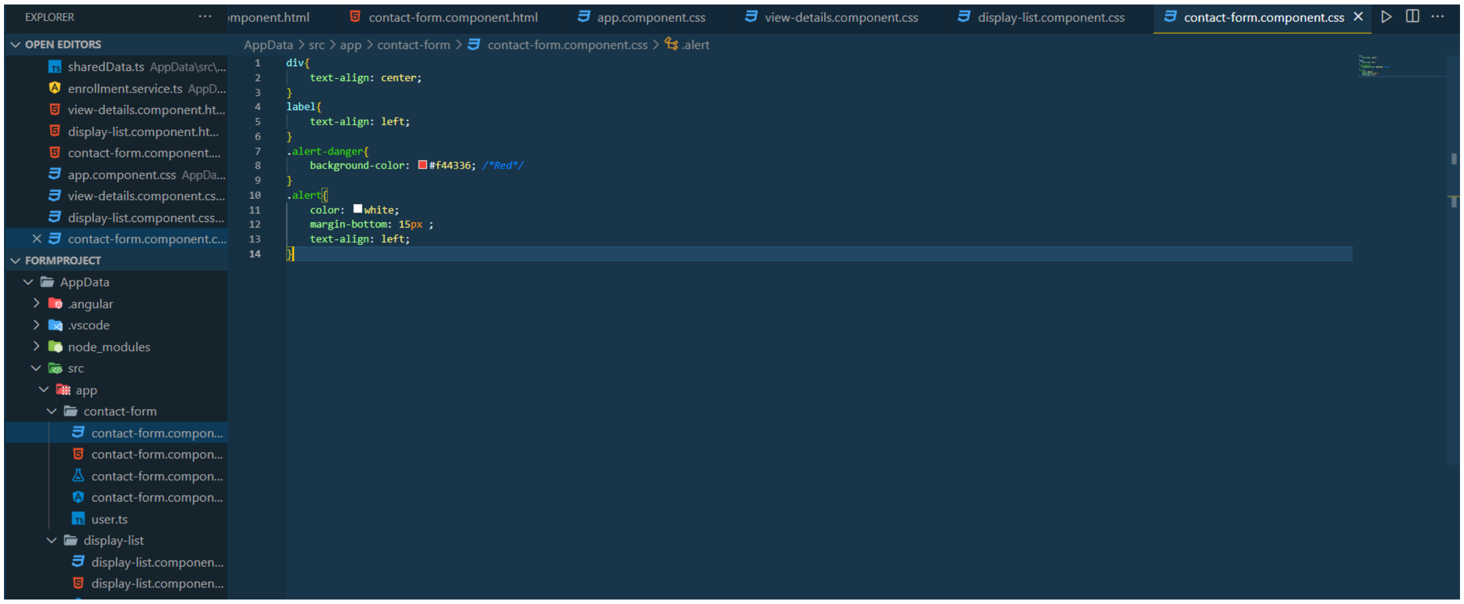
```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

√ Compiled successfully.
```


OPEN EDITORS
FORMPROJECT
OUTLINE
AppData > src > app > contact-form > contact-form.component.html > div.container > form > div.form-group

```html
        <hr>
        <br>
        <div class="form-group">
            <label for="firstname">First Name</label><br>
            <input  type="text" #name="ngModel" [class.is-invalid]="name.invalid && name.touched" required name="firstname" class="form-control" [(ngModel)]="userModel.firstName">
            <!-- <small class="text-danger" [class.d-none]="name.valid || name.untouched">Name is required</small> -->
            <!-- <div *ngif="name.touched && name.invalid" class="alert alert-danger">
                <p *ngIf="name.errors.required">FirstName Required!!</p>
                <p *ngIf="name.errors.minlength">Sorry! Short FirstName</p>
            </div> -->
        </div>
        <pre></pre>
        <div class="form-group">
            <label for="lastname">Last Name</label><br>
            <input type="text" name="lastname" class="form-control" [(ngModel)]="userModel.lastName">
        </div>
        <pre></pre>
        <div class="form-group">
            <label for="email">Email</label><br>
            <input type="text" name="email" class="form-control" [(ngModel)]="userModel.email">
        </div>
        <pre></pre>
        <div class="form-group">
            <label for="password">Password</label><br>
            <input type="text" #password="ngModel"   [class.is-invalid]="password.invalid && password.touched" name="password" class="form-control" [(ngModel)]="userModel.
            password">
        </div>
        <pre></pre>
        <button  type="submit" [disabled]="login.form.invalid"  mat-raised-button color="warn" type="">Submit</button>
        <hr>
    </form>
</div>
```

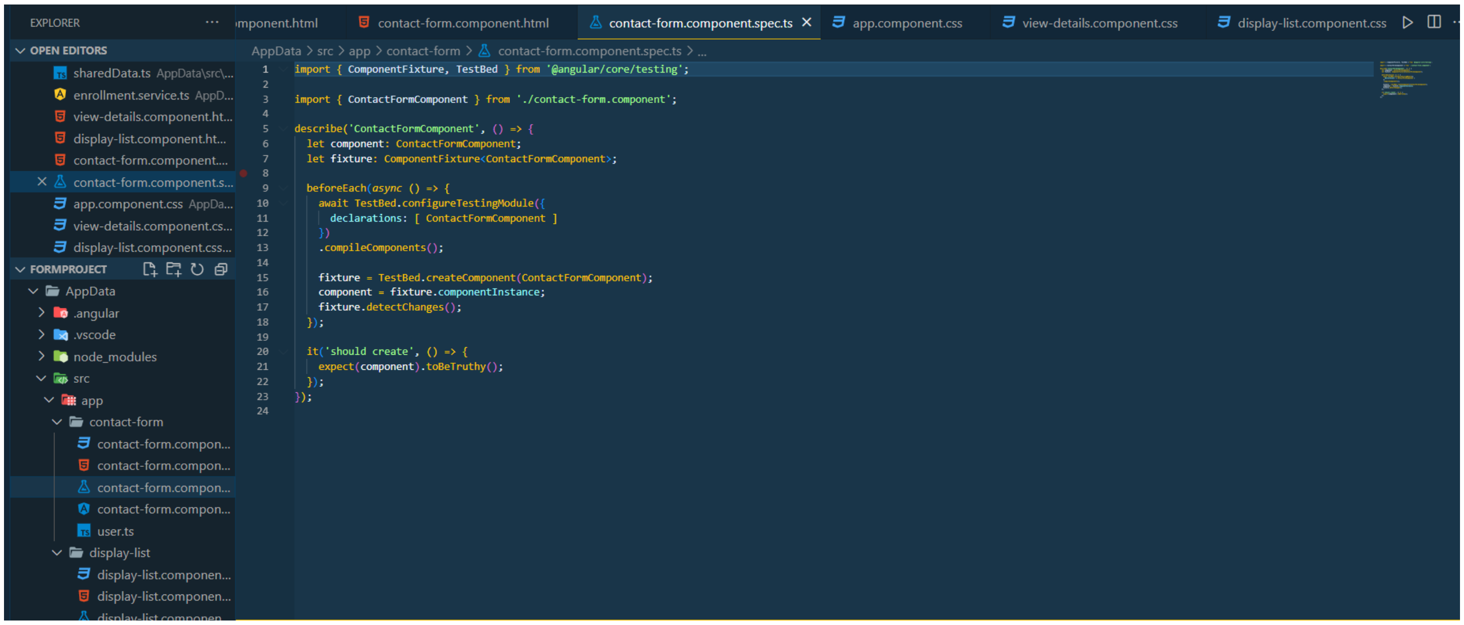
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
node

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
```



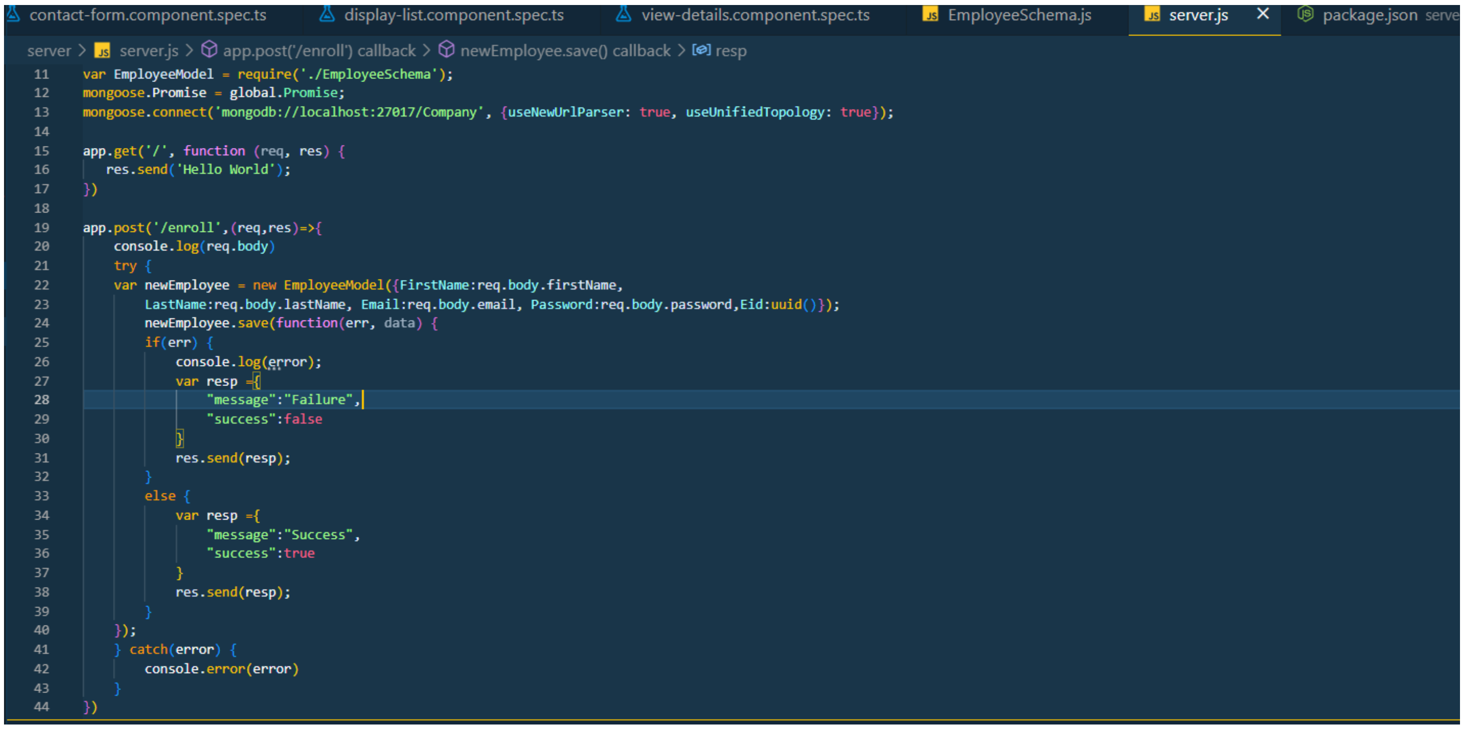
AppData > src > app > contact-form > contact-form.component.css > .alert

```css
div{
    text-align: center;
}
label{
    text-align: left;
}
.alert-danger{
    background-color: #f44336; /*Red*/
}
.alert{
    color: white;
    margin-bottom: 15px ;
    text-align: left;
}
```

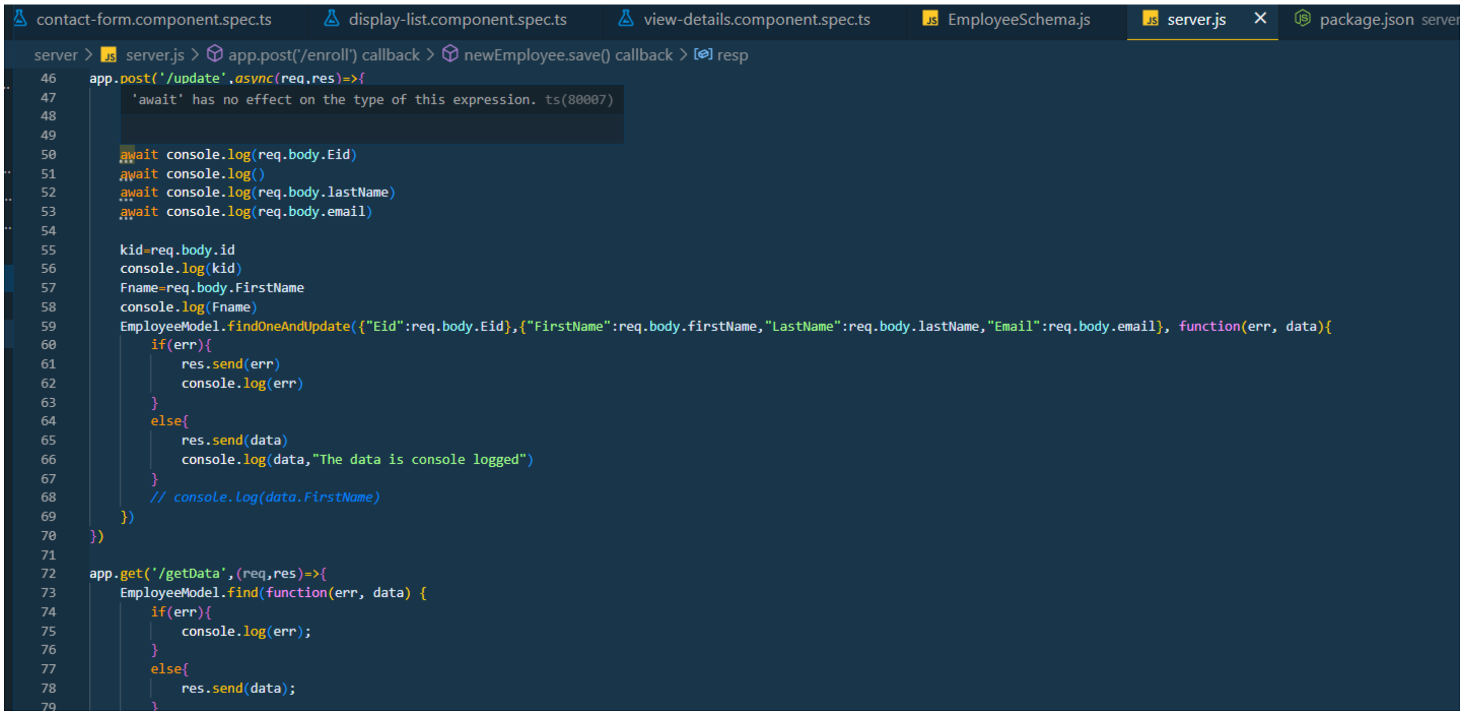


AppData > src > app > contact-form > contact-form.component.spec.ts > ...

```typescript
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ContactFormComponent } from './contact-form.component';

describe('ContactFormComponent', () => {
  let component: ContactFormComponent;
  let fixture: ComponentFixture<ContactFormComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ContactFormComponent ]
    })
    .compileComponents();

    fixture = TestBed.createComponent(ContactFormComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

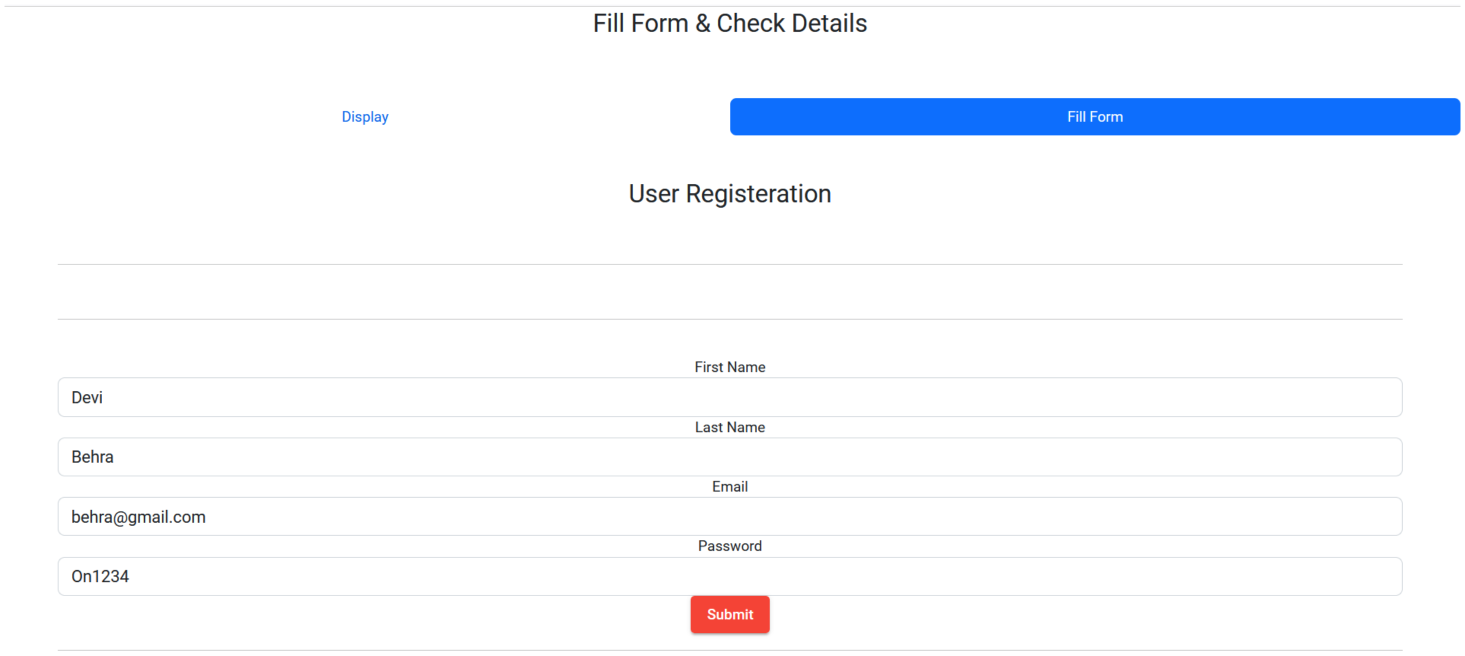

contact-form.component.spec.ts
display-list.component.spec.ts
view-details.component.spec.ts
EmployeeSchema.js
server.js
package.json
server > server.js > app.post('/enroll') callback > newEmployee.save() callback > resp

```javascript
var EmployeeModel = require('./EmployeeSchema');
mongoose.Promise = global.Promise;
mongoose.connect('mongodb://localhost:27017/Company', {useNewUrlParser: true, useUnifiedTopology: true});

app.get('/', function (req, res) {
   res.send('Hello World');
})

app.post('/enroll',(req,res)=>{
    console.log(req.body)
    try {
    var newEmployee = new EmployeeModel({FirstName:req.body.firstName,
        LastName:req.body.lastName, Email:req.body.email, Password:req.body.password,Eid:uuid()});
        newEmployee.save(function(err, data) {
        if(err) {
            console.log(error);
            var resp ={
                "message":"Failure",
                "success":false
            }
            res.send(resp);
        }
        else {
            var resp ={
                "message":"Success",
                "success":true
            }
            res.send(resp);
        }
    });
    } catch(error) {
        console.error(error)
    }
})
```


contact-form.component.spec.ts
display-list.component.spec.ts
view-details.component.spec.ts
EmployeeSchema.js
server.js
package.json
server > server.js > app.post('/enroll') callback > newEmployee.save() callback > resp

```javascript
app.post('/update',async(req,res)=>{
        'await' has no effect on the type of this expression. ts(80007)


        await console.log(req.body.Eid)
        await console.log()
        await console.log(req.body.lastName)
        await console.log(req.body.email)

        kid=req.body.id
        console.log(kid)
        Fname=req.body.FirstName
        console.log(Fname)
        EmployeeModel.findOneAndUpdate({"Eid":req.body.Eid},{"FirstName":req.body.firstName,"LastName":req.body.lastName,"Email":req.body.email}, function(err, data){
            if(err){
                res.send(err)
                console.log(err)
            }
            else{
                res.send(data)
                console.log(data,"The data is console logged")
            }
            // console.log(data.FirstName)
        })
})

app.get('/getData',(req,res)=>{
        EmployeeModel.find(function(err, data) {
            if(err){
                console.log(err);
            }
            else{
                res.send(data);
            }
```

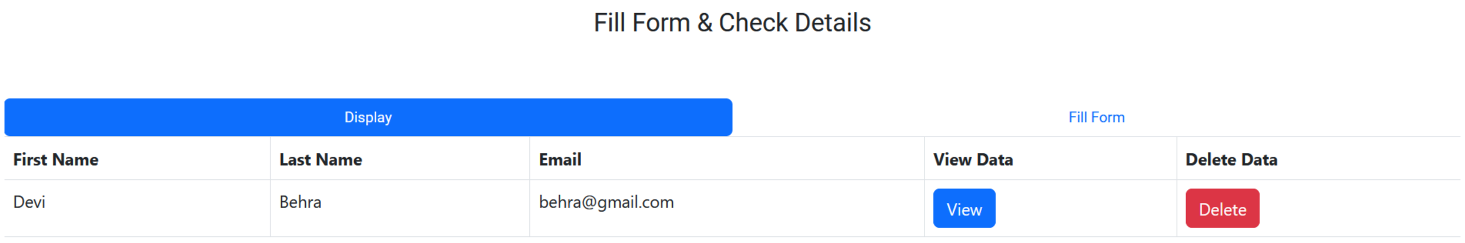
OUTPUT:-



Fill Form & Check Details

User Registeration

First Name	Devi
Last Name	Behra
Email	behra@gmail.com
Password	On1234



Fill Form & Check Details

First Name	Last Name	Email	View Data	Delete Data
Devi	Behra	behra@gmail.com	View	Delete

CONCLUSION

In conclusion, the quest to conquer the manual method of filling forms has been turned into an online experience.

REFERENCES

- Educator and learning experience : https://www.javatpoint.com/
- For angular docs; https://angular.io/docs
- For node.js : https://nodejs.org/
- https://www.w3schools.com/

COMPANY DETAILS:-

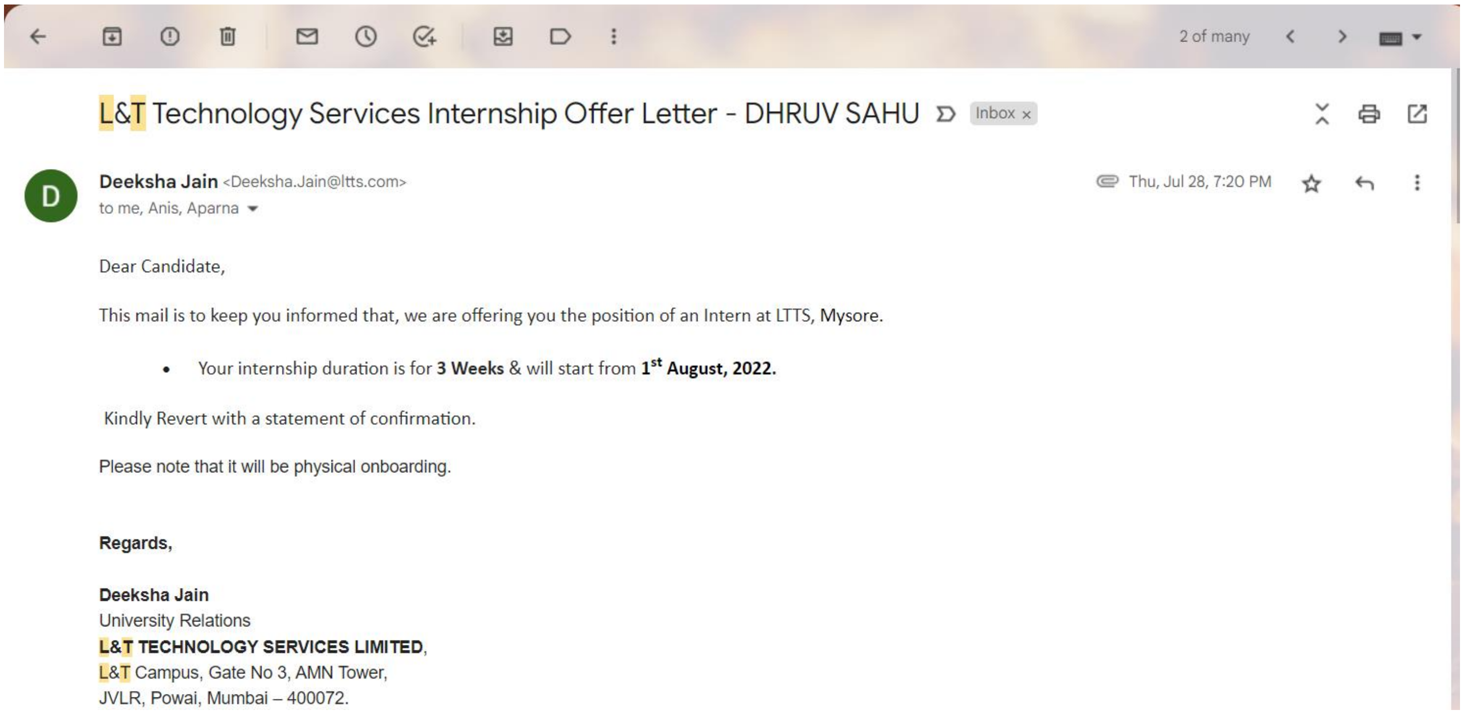
NAME:- Larsen and Tubro

LOCATION:- KIADB Industrial Area Hebbal, Hootagalli, Mysuru, Karnataka 570018

HR:- Deeksha Jain

EMAIL-ID:- Deeksha.Jain@ltts.com

WEBSITE:- https://www.ltts.com/


2 of many

L&T Technology Services Internship Offer Letter - DHRUV SAHU Inbox ×

Deeksha Jain <Deeksha.Jain@ltts.com>
to me, Anis, Aparna

Thu, Jul 28, 7:20 PM

Dear Candidate,

This mail is to keep you informed that, we are offering you the position of an Intern at LTTS, Mysore.

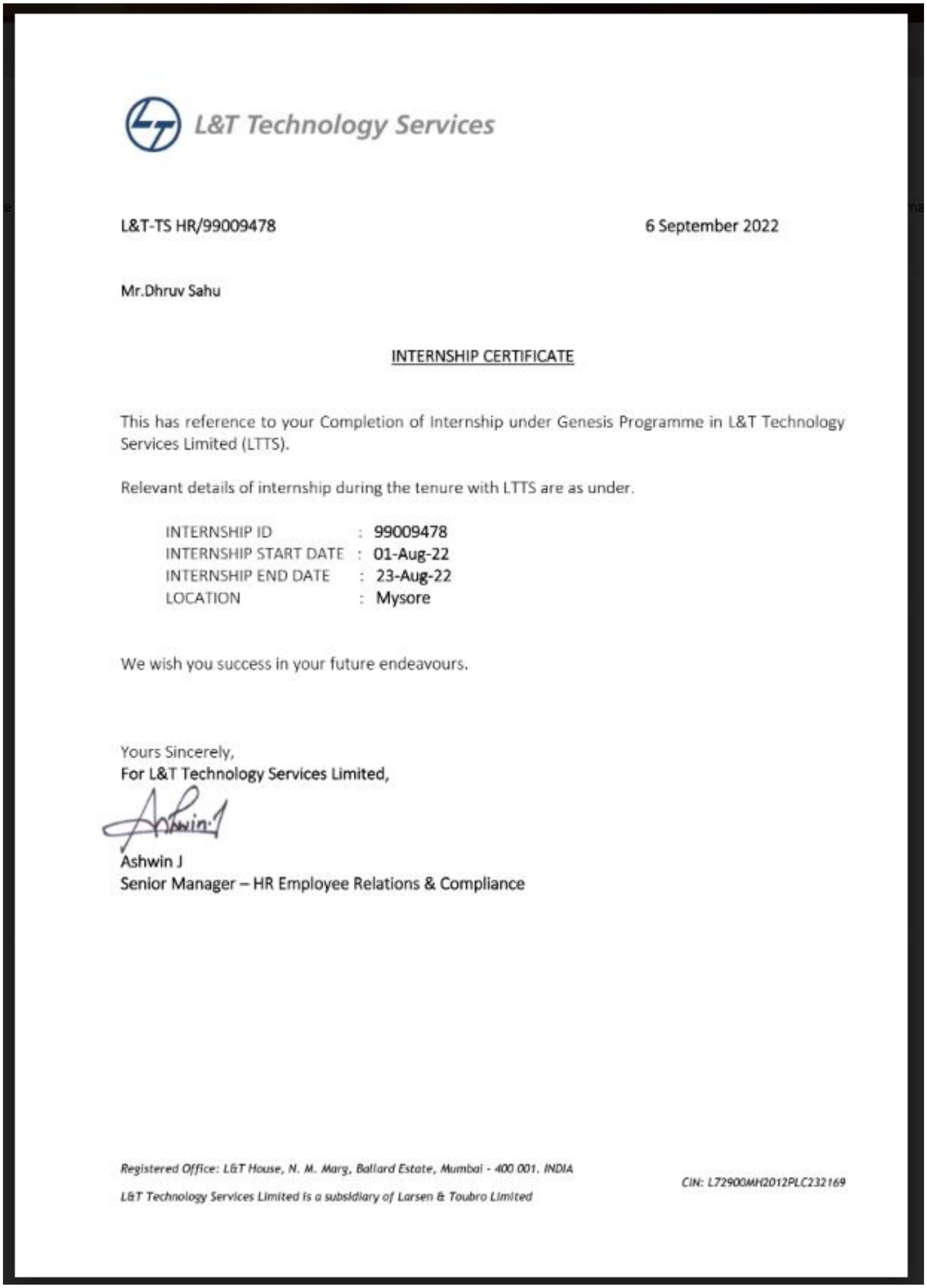
- Your internship duration is for **3 Weeks** & will start from **1st August, 2022.**

Kindly Revert with a statement of confirmation.

Please note that it will be physical onboarding.

Regards,

Deeksha Jain
University Relations
L&T TECHNOLOGY SERVICES LIMITED,
L&T Campus, Gate No 3, AMN Tower,
JVLR, Powai, Mumbai – 400072.



L&T Technology Services

L&T-TS HR/99009478

6 September 2022

Mr.Dhruv Sahu

INTERNSHIP CERTIFICATE

This has reference to your Completion of Internship under Genesis Programme in L&T Technology Services Limited (LTTS).

Relevant details of internship during the tenure with LTTS are as under.

INTERNSHIP ID	:	99009478
INTERNSHIP START DATE	:	01-Aug-22
INTERNSHIP END DATE	:	23-Aug-22
LOCATION	:	Mysore

We wish you success in your future endeavours.

Yours Sincerely,
For L&T Technology Services Limited,

[illegible]

Ashwin J
Senior Manager – HR Employee Relations & Compliance