

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology (Deemed to be University)

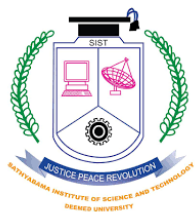
Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Information Technology

By
Abhinav Kumar
REG. NO. 39120003



DEPARTMENT OF Information Technology
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMIL NADU

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE
UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

www.sathyabama.ac.in



DEPARTMENT OF Information Technology

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Abhinav Kumar (Reg. No: 39120003)** who carried out the project entitled “**ONLINE CHARITY MANAGEMENT SYSTEM**” under my supervision from June 2021 to November 2021.

Internal Guide

Dr.P.Jeyanthi

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, Abhinav Kumar hereby declare that the project report entitled **Online charity management** done by me under the guidance of Dr. P. Jeyanthi is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to the Board of management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to, **Dr. T. SASIKALA M.E., Ph.D.**, Dean, School of Computing , **Dr.R.Subhasini**, Heads of the Department, Department of Information Technology for providing me the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.P.Jeyanthi** for her valuable guidance, suggestions which paved way for the successful completion of my project work.

I wish to express my thanks to all teaching and non-teaching staff members of the department of Information Technology who were helpful in many ways for the completion of project.

TRAINING CERTIFICATE

CERTIFICATE OF EXCELLENCE

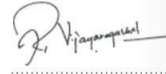
THIS IS TO CERTIFY THAT

Abhinav Kumar

has successfully completed
ANGULAR CERTIFIED TRAINING
Course Conducted During September 2021 - November 2021

Date: 07-11-2021




Authorised Signatory

ABSTRACT

Online charity management system is a website through which user can donate. This system provides a solution using JavaScript Frameworks like Angular, Nodejs, express and database like MONGODB.

Angular is a platform and framework for building single page application using html and typescript. It implements core and optional functionality as a set of typescript libraries that we import into our application. The architecture of an angular application relies on certain fundamental concepts. The basic building blocks are the Ng modules. In this project, many components are created and routes are set up between each component for the dynamic motion of one page to another page. In this charity app, the user can view the details of the active charity details, the admin can add and remove charity details, add, edit and delete users' details also. Another feature of the app is that the admin user can search the saved user detail.

For the backend, the app is connected to the mongo dB so that information of the charity and logged in users will be stored and displayed.

TABLE OF CONTENTS

Chapter No.	TITLE	Page
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1.	INTRODUCTION	10
	WHAT IS ANGULAR JS?	10
	ROLE OF HTML, CSS & JS IN ANGULAR	11
	REQUIREMENTS FOR ANGULAR PROJECT	12
2.	Online Charity Management System APP	13
	OVERVIEW	13
	FUNCTIONAL REQUIREMENTS	13
COMPONENTS		14
	OVERVIEW OF COMPONENTS	14
	BACKEND	21
3.	SUMMARY AND CONCLUSIONS	22
	SUMMARY	
	CONCLUSIONS	
	Reference	23
	APPENDIX	23
	A. SOURCE CODE	23
	B. SCREENSHOTS	49

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
1.1	Login Page	15
1.2	Register Page	15
1.3	User charity detail	16
1.4	User Management Page	18
1.4.1	Add charity detail	18
1.4.2	Edit charity detail	19
1.4.3	Delete charity detail	20
1.5	Mongo dB connection	22

LIST OF TABLES

TABLE No.	TITLE	PAGE No.
2.1	Login Fields	15
2.2	Register/Sign-up Fields	15
2.3	user management detail	18
2.5	Add charity detail	18
2.6	Edit charity detail	19

LIST OF ABBREVIATIONS

Mongo dB	-	Mongo database
App	-	Application
Chars	-	Characters
No's	-	Numbers
Acc	-	According
JS	-	Java Script
Const	-	constant

INTRODUCTION

What is Angular?

The Angular framework works by first reading the Hypertext Mark-up Language (HTML) page, which has additional custom HTML attributes embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

Angular is built on the belief that declarative programming should be used to create user interfaces and connect software components, while imperative programming is better suited to defining an application's business logic. The framework adapts and extends traditional HTML to present dynamic content through two-way data-binding that allows for the automatic synchronization of models and views. As a result, AngularJS de-emphasizes explicit Document Object Model (DOM) manipulation with the goal of improving testability and performance.

Angular design goals include:

- To decouple DOM manipulation from application logic. The difficulty of this is dramatically affected by the way the code is structured.
- To decouple the client side of an application from the server-side. This allows development work to progress in parallel and allows for reuse of both sides.
- To provide structure for the journey of building an application: from designing the UI, through writing the business logic, to testing.

Angular implements the MVC pattern to separate presentation, data, and logic components. Using dependency injection, Angular brings traditionally server-side services, such as view-dependent controllers, to client-side web applications. Consequently, much of the burden on the server can be reduced. Angular directives allow the developer to specify custom and reusable HTML-like elements and attributes that define data bindings and the behaviour of presentation components. Some of the most commonly used directives are:

Since **ng-*** attributes are not valid in HTML specifications, **data-ng-*** can also be used as a prefix. For example, both **ng-app** and **data-ng-app** are valid in AngularJS.

ng-app

Declares the root element of an Angular application, under which directives can be used to declare bindings and define behavior.

ng-aria

A module for accessibility support of common ARIA attributes.

ng-bind

Sets the text of a DOM element to the value of an expression. For example, `<span ng-bind="name"></span>` displays the value of 'name' inside the span element. Any change to the variable 'name' in the application's scope reflect instantly in the DOM.

ng-class

Conditionally apply a class, depending on the value of a Boolean expression.

ng-controller

Specifies a JavaScript controller class that evaluates HTML expressions.

ng-if

Basic if statement directive that instantiates the following element if the conditions are true. When the condition is false, the element is removed from the DOM. When true, a clone of the compiled element is re-inserted.

ng-init

Called once when the element is initialized.

ng-model

Similar to **ng-bind**, but establishes a two-way data binding between the view and the scope.

Role of HTML, CSS & JS in Angular:

- A webpage can contain headings, paragraphs, images, videos, and many other types of data. Front end developers use the HTML element to specify what kind of information each item on a webpage contains — for instance, the “p” HTML element indicates a paragraph. Developers also write HTML language code to specify how different items relate to one another in the page’s overall structure or document structure.
- CSS is used for defining the styles for web pages. It describes the look and formatting of a document which is written in a markup language. It provides an additional feature to HTML. ... CSS helps us to control the text color, font style, the spacing between paragraphs, sizing of columns, layout designs, and many more. CSS provides these capabilities through a technical model based on a hierarchical scope of effect, the separation of style from content, and a well-defined set of published standards.
- JavaScript is the most popular and most widely developed programming language ever existed. JavaScript is a must-have for Web developers, Front-end/ Back-end developers, Mobile / Desktop developers, Graphics / Game developers and even for Full Stack developers as well.
- Angular is a platform and framework for building single-page client applications using HTML and Typescript. Angular is written in Typescript. It implements core and optional functionality as a set of Typescript libraries that you import into your applications.
- AngularJS provides some amazing capabilities out of the box, and can convert your non-trivial jQuery based apps into something more powerful and yet more simple to maintain. Some frameworks simply bundle together the existing tools which make app development very difficult. However, Angular was carefully engineered to ensure that every tool works optimally and delivers great results.
- Developers find Angular very effective in terms of creating dynamic, single page application, and supporting MVC (Model-View-Controller) programming structure. At a present span of time, in the competitive business environment, Angular has emerged as the superhero due to its intuitiveness and rich feature

Requirements of Angular Project:

- Install Angular CLI
- Create a workspace and initial application
- Run the application

Modules, components and services are classes that use *decorators*. These decorators mark their type and provide metadata that tells Angular how to use them.

- The metadata for a component class associates it with a *template* that defines a view. A template combines ordinary HTML with Angular *directives* and *binding mark-up* that allow Angular to modify the HTML before rendering it for display.
- The metadata for a service class provides the information Angular needs to make it available to components through *dependency injection (DI)*.

Angular *NgModules* differ from and complement JavaScript (ES2015) modules. An *NgModule* declares a compilation context for a set of components that is dedicated to an application domain, a workflow, or a closely related set of capabilities. An *NgModule* can associate its components with related code. Every Angular application has at least one component, the *root component* that connects a component hierarchy with the page document object model (DOM). Each component defines a class that contains application data and logic, and is associated with an HTML *template* that defines a view to be displayed in a target environment.

The `@Component ()` decorator identifies the class immediately below it as a component, and provides the template and related component-specific metadata.

A template combines HTML with Angular mark-up that can modify HTML elements before they are displayed. Template *directives* provide program logic, and *binding mark-up* connects your application data and the DOM.

ONLINE CHARITY APPLICATION

Overview:

This project aims to maintain the information about the charities for those who are affected by poverty. The admin will manage details of the charities. Admin will be able to login themselves and add data of the charities and for the admin purpose, user management portal will be also available.

Only admin will be able to update charities information. There are many components like login, charity details etc. Once user register themselves, they will be directed to Home page. This app is done with the help of Angular Framework.

Our Application design is very easy and user friendly for the visitors to view the details of affected people. Now a days, internet is the most widely used network in modern world , For their easy purpose, Team Flames made this charity App, so that users can acknowledge more information about the people seeking for help around them.

Functional Requirements:

To build an angular application, the main thing needed is the components where are those application routing process occurs. The components used in this application are listed below:

COMPONENTS:

- ❖ Login
- ❖ Register
- ❖ Charities details
- ❖ User Management Actions (for admin purpose)
 - Add
 - Edit
 - Search
 - Delete
- ❖ Logout

OVERVIEW OF COMPONENTS:

- **Login portal:**

The login portal is used for the users to login into the app. The admin also can login to do the user management system.

5

FIELDS	VALIDATIONS
user name	Required
Password	Required

Table 2.1: Login Fields

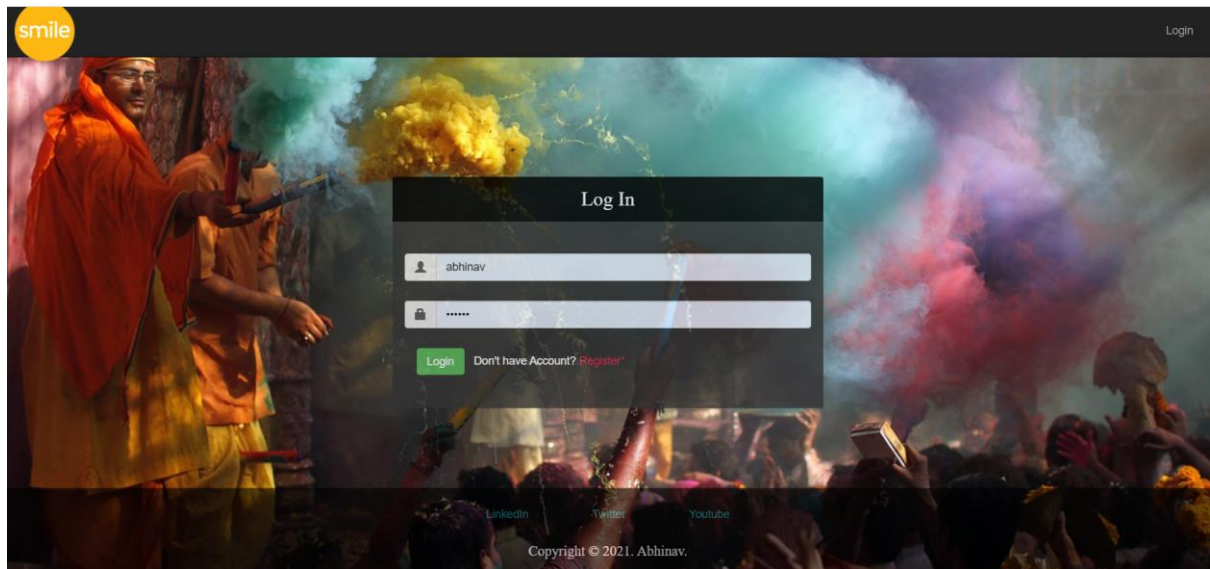


Fig 1.1: Login Page

- **Register:**

User will be able to register themselves by clicking register link from login page. On submission, data gets inserted to the database via mongo db.

FIELDS	VALIDATIONS
Username	Required
E-mail	Required, E-mail validation
Password	Required

Phone no	Number validation,
----------	--------------------

Table 2.2: Register

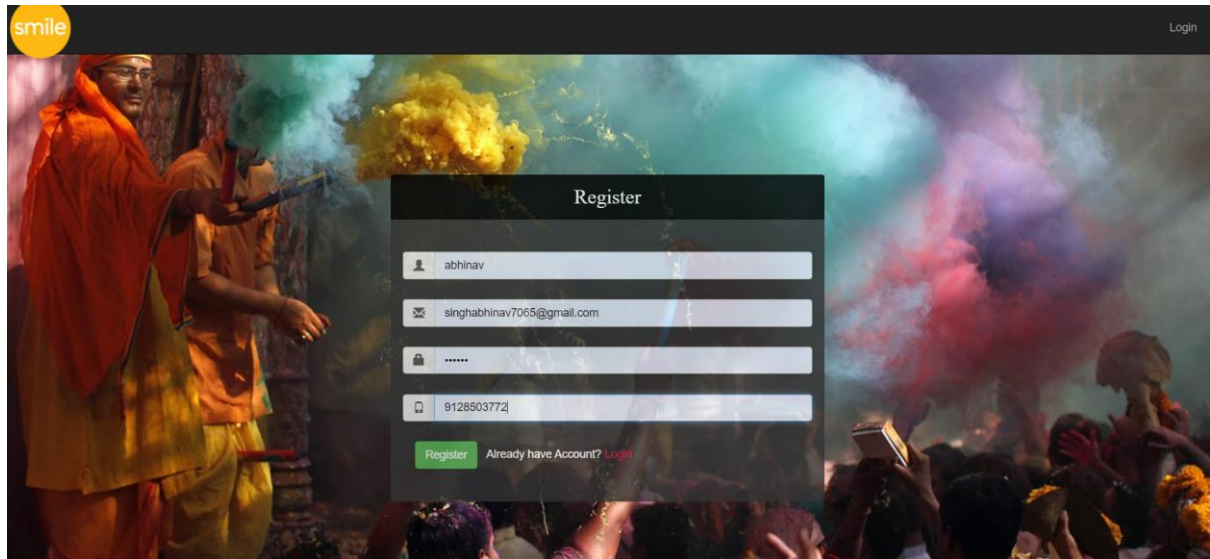


Fig 1.2: Register Page

7

- **Charity details:**

In Charity detail portal, the logged in users can view all the active charities details.

FIELDS	VALIDATIONS
username	Required
DOB	Required
Email	Required
Address	Required

Table 2.3: Charity details

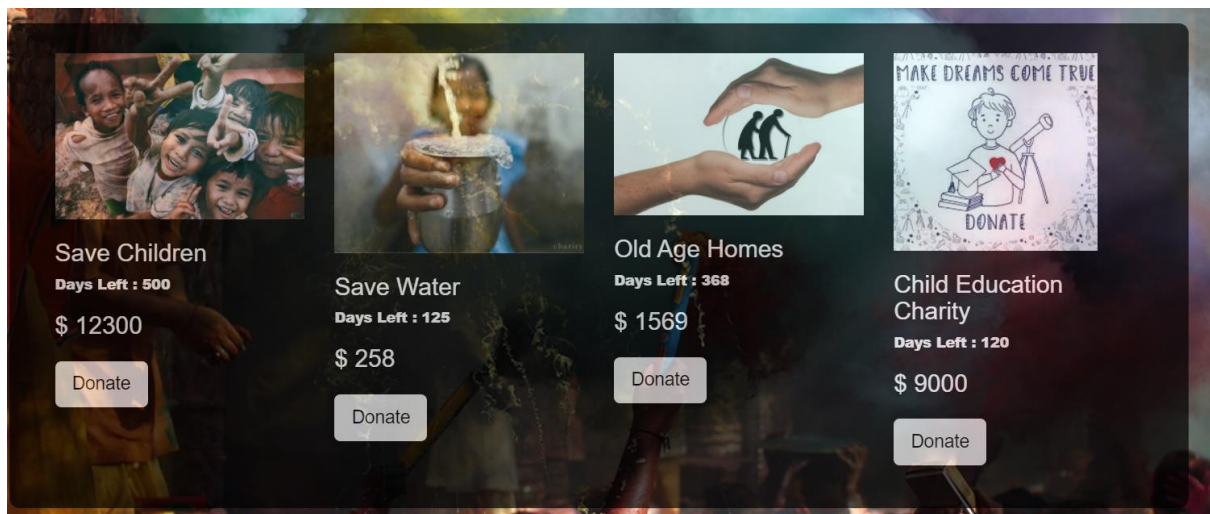


Fig 1.3: Charity details

- **User Management Portal:**

This portal is fully for the purpose of admin in-order to add, edit and delete the details from the management portal.

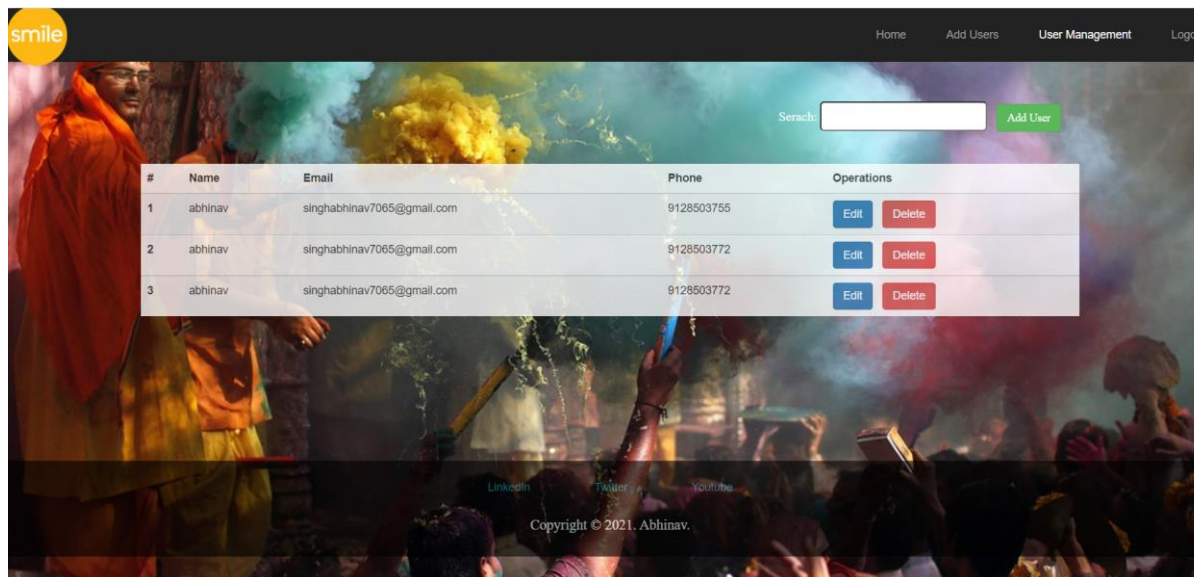


Fig 1.6: User Management Page

- There are certain actions like add, edit and delete info from database.

- **Add Charity Portal:**

Add Charity component mainly involves the admin action of adding charity.

FIELDS	VALIDATIONS
Image	Jpg file
Beneficiary name	Required,
Amount to be raised	Required
Days left	Required

Table 2.5: Add New charity Fields

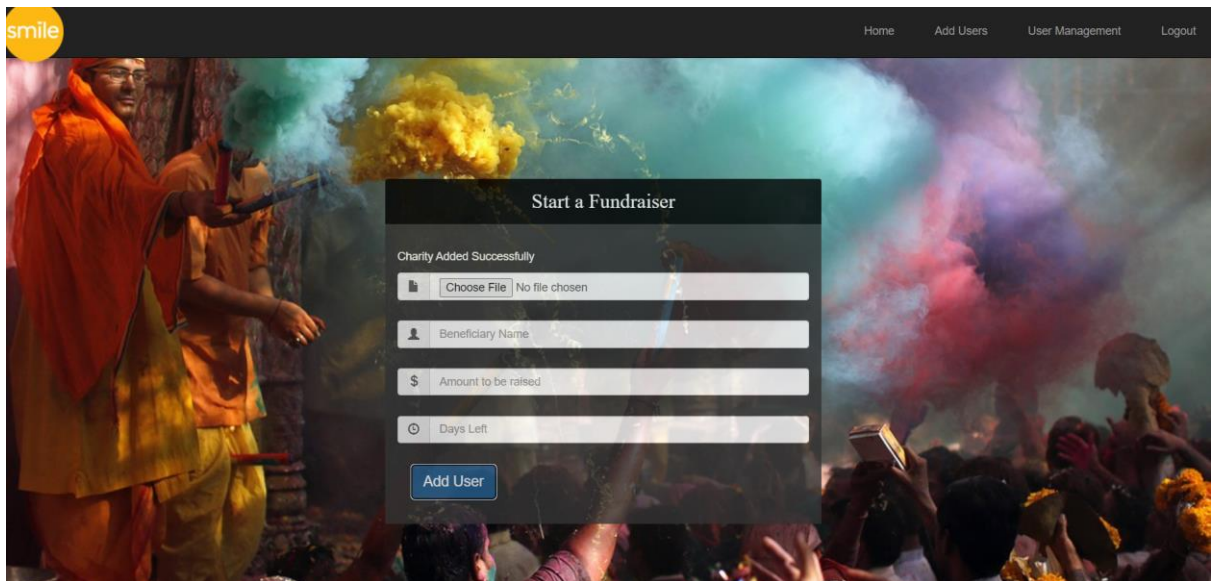


Fig 1.6.1: Add New Charity Page

- **Edit user's Portal:**

The user management portal contains a button named edit in side of every registered user's name. On clicking that button the admin gets directed to the edit user portal where all the data of user's already entered will be available. Thereby admin can update the user's information and the database also gets updated.

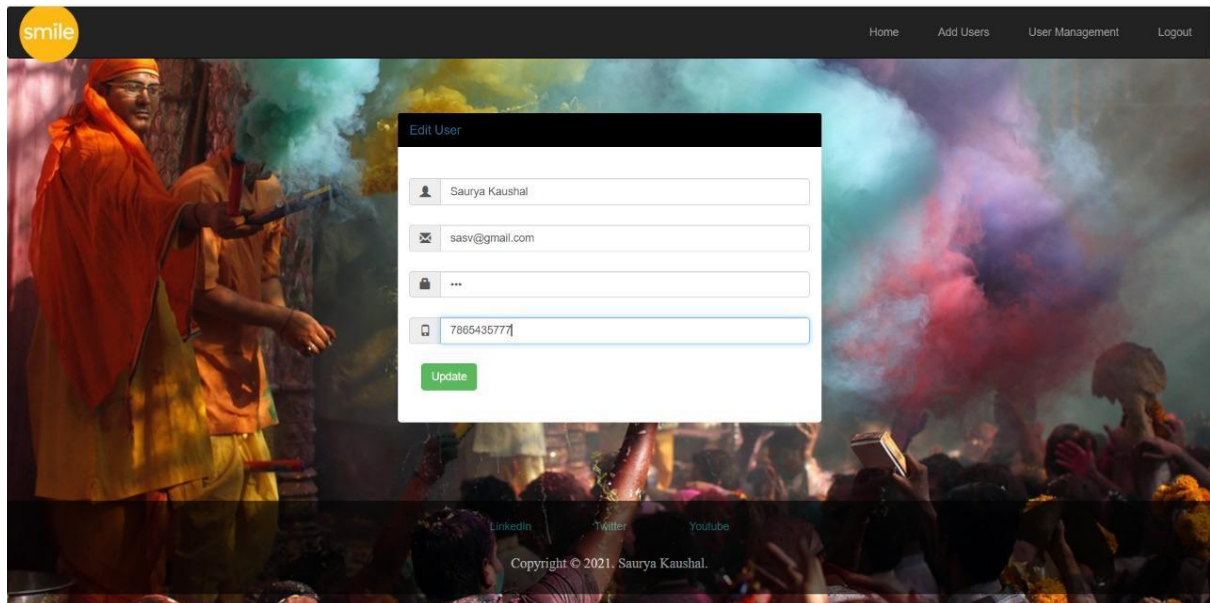


Fig 1.6.2: Edit user's portal

- **Delete User Option:**

Same like edit option, there exists a delete button in side of every user's data. On clicking that delete button, a dialogue box appears asking that sure deletion should be done or not. As per the admin wish, the deletion operation will be done.

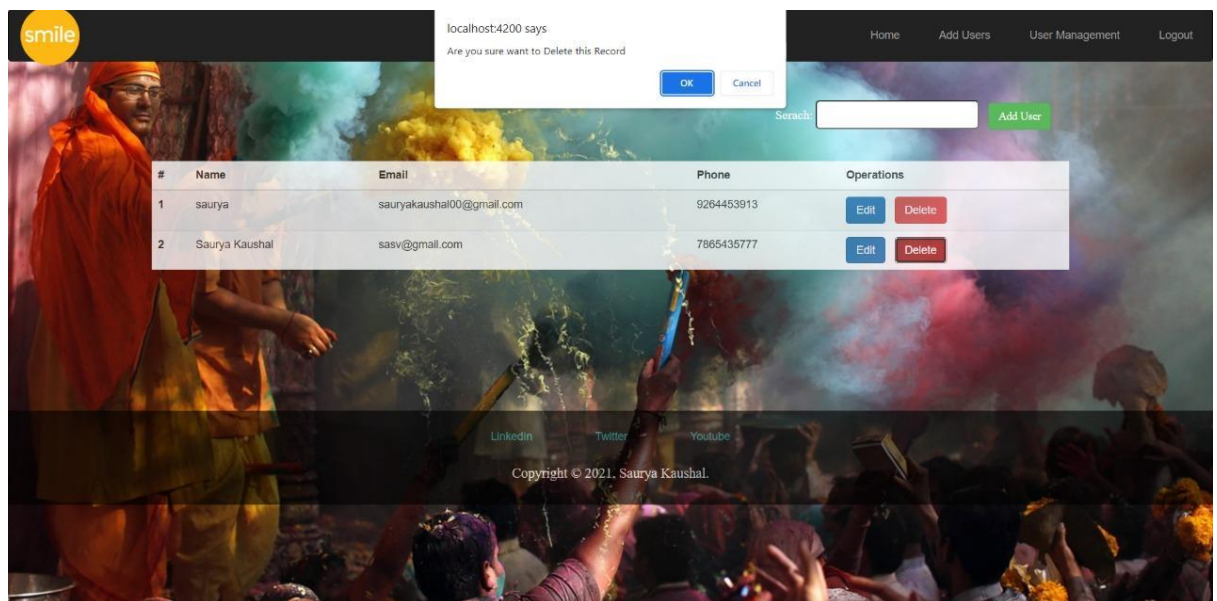


Fig 1.6.4: Delete user Page

- **Logout:**

After updating everything, the info will be updated in user portal .once the update is completed the admin can logout.
This is the final stage of our application.

- **BACKEND:**

The backend duties involve the mongo dB connection and a service. For the backend actions, a java script document is done in backend and thereby the mongo dB cluster is connected to the application.

For this to be carried out, first of all a mongo dB cluster should be created and a database should be made and thereby it should be connected with backend through the mongo dB database connection link.

A service named user service is imported to carry out all the backend function and call back functions made to get the responses of the server.

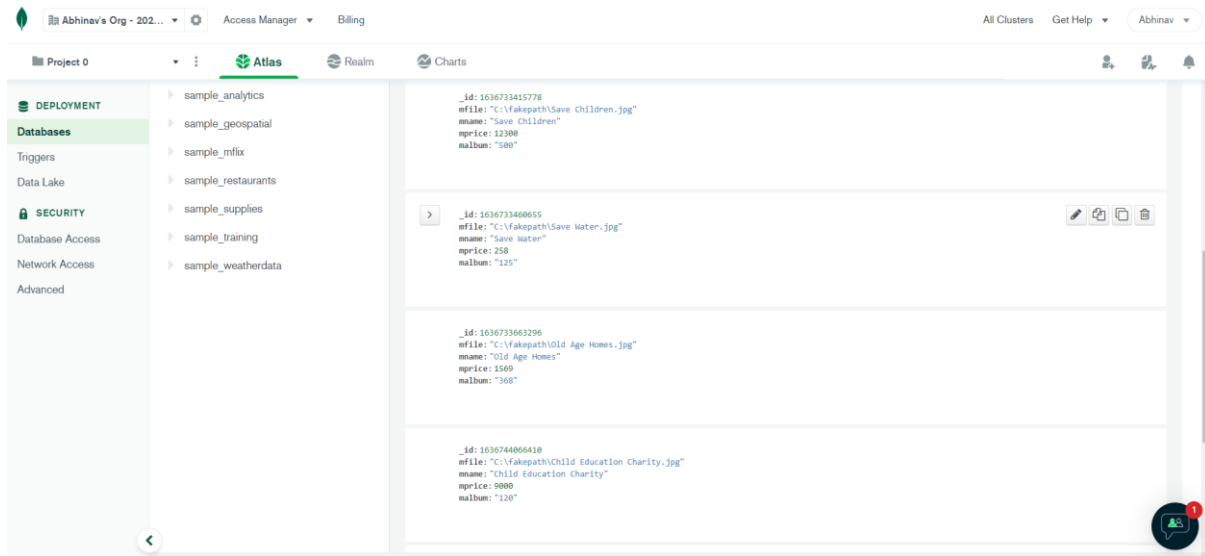


Fig 1.8: Mongo dB Connection

SUMMARY AND CONCLUSIONS

Online Charity web apps like Smile Charities are not just a change of technology. It is part of a redefinition of how we as a species, sitting at our home, can help the needy ones miles of away from us. The software ‘Online charity store’ reduces the considerable drawbacks like burden of human labor, portable defect and errors. This software saves time and provides 24 hour accessibility even from a remote place. Programs are menu driven which help even a newcomer to use the system with little training. Testing has been done with actual data and system is much better than the existing one. CSS makes the interface very much user friendly.

The system is highly user friendly and is well efficient to ease interactions with the users of the system. Reports generate with live data are proved to be informative and also helpful in making important decisions. The system is tested and implemented with high degree

of accuracy.

The system is done with an insight into the necessary modification that may require in the future. Hence the system can be maintained successfully, without much rework.

REFERENCES

1. www.google.com
2. <https://developer.mozilla.org/en-US/>
3. www.pexels.com
5. www.wikipedia.org/angular
6. www.tutorialspointsg.com
7. www.javaworld.com
8. www.angularjs.com
9. www.nodejs.com

APPENDIX

Source Code

(1). Backend

```
const express = require("express");
const cors = require("cors");
const bodyparser = require("body-parser");
const mongodb = require("mongodb").MongoClient;

const app = express();

app.use(cors());

app.use(bodyparser.json());

var db;
```

```

mongodb.connect("mongodb+srv://sk1102:SauryaKaushal@cluster0.m4slg.mongodb.net/charityDetails?retryWrites=true&w=majority",(error, result)=>{

if(error)
{
    console.log("DB Not Connect");
}
else
{
    db = result.db("collegebatch1");
    console.log("DB Connected");
}

});

app.use((req, res, next)=>{                //middleware for all paths

    console.log("Middleware 1");

    next();
});

app.use("/home", (req, res, next)=>{        // middleware for /home incoming
request path

    console.log("Middleware 2");

    next();
});

function verifyUser(req, res, next)
{
    console.log("User Verified");

    next();
}

app.get("/", (req, res)=>{

    console.log("Index Page");

    //res.send("<h1>Welcome to Express JS</h1>");

    var data = { a:"hi", b:"hello"};

    res.json(data);

});

app.get("/home", verifyUser, (req, res)=>{

    console.log("Home Page");

    res.json("Welcome to our homepage");

});

```

```

app.post("/register", (req, res) => {
    req.body._id = new Date().getTime();
    console.log(req.body);
    db.collection("users").save(req.body, (error, data) => {
        if (error) {
            res.status(403).json("Error in insert Query");
        }
        else {
            res.json("User Registered Successfully");
        }
    });
});

app.post("/login", (req, res) => {
    console.log(req.body);
    db.collection("users").find(req.body, {projection: {_id : 1, uname : 1}}).toArray((error, data) => {
        if (error) {
            res.status(403).json("Error in finding the login document");
        }
        else {
            res.json(data);
        }
    });
});

app.get("/allusers", (req, res) => {
    db.collection("users").find(null, {projection: {upassword : 0}}).toArray((error, data) => {
        if (error) {
            res.status(403).json("Error in finding the login document");
        }
        else {
            res.json(data);
        }
    });
});

app.get("/usernamecheck/:username", (req, res) => {

```

```

    console.log(req.params.username);

    db.collection("users").find({uname: req.params.username}, {projection:{_id
: 1}}).toArray((error, data)=>{

        if(error)
        {
            res.status(403).json("Error in finding Username Availability");
        }
        else
        {
            res.json(data);
        }
    });
});

app.get("/getuser/:userid", (req, res)=>{

    console.log(req.params);

    db.collection("users").find({_id :
Number(req.params.userid)}).toArray((error, data)=>{

        if(error)
        {
            res.status(403).json("Error in finding the Document");
        }
        else
        {
            res.json(data);
        }
    });
});

app.put("/updateuser",(req, res)=>{

    console.log(req.body);

    var condition = {_id : req.body._id};

    var newValues ={$set: {uname: req.body.uname, uemail: req.body.uemail,
upassword: req.body.upassword, uphone: req.body.uphone}};

    db.collection("users").update(condition, newValues, (error, data)=>{

        if(error)
        {
            res.status(403).json("Error in Updating the Document");
        }
        else
        {
            res.json("User Updated Successfully");
        }
    })
});
});

```

```

app.delete("/deleteuser/:userid",(req, res)=>{

    console.log(req.params);

    db.collection("users").deleteOne({_id : Number(req.params.userid)},(error,
data)=>{

        res.json("User Deleted Successfully");

    });
});

app.get("/search/:searchtxt?", (req, res)=>{

    console.log(req.params);

    if(req.params.searchtxt!=undefined)
    {
        var search = new RegExp(req.params.searchtxt, 'i');
        var searchCond = {uname : search};
    }
    else
    {
        var searchCond = null;
    }

    db.collection("users").find({searchCond}).toArray((error, data)=>{

        res.json(data);

    });
});

app.post("/addcharity",(req, res)=>{

    req.body._id = new Date().getTime();

    console.log(req.body);

    db.collection("charity").save(req.body, (error, data)=>{

        if(error)
        {
            res.status(403).json("Error in insert Query");
        }
        else
        {
            res.json("New Charity Details Added Successfully");
        }
    });
});
});

```

```

app.get("/allcharity", (req, res)=>{

    db.collection("charity").find().toArray((error, data)=>{

        if(error)
        {
            res.status(403).json("Error in finding the document");
        }
        else
        {
            res.json(data);
        }
    });
});

module.exports = app;

```

(2.) ANGULAR.JSON

```

{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "cli": {
    "analytics": "d89cb836-b026-4d37-b07f-b76e98a74a39"
  },
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "firstProject": {
      "projectType": "application",
      "schematics": {
        "@schematics/angular:application": {
          "strict": true
        }
      },
      "root": "",
      "sourceRoot": "src",
      "prefix": "app",
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/firstProject",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets"
            ],
            "styles": [
              "src/assets/css/bootstrap.css",
              "src/styles.css"
            ],
            "scripts": []
          }
        }
      }
    }
  }
}

```

```

    },
    "configurations": {
      "production": {
        "budgets": [
          {
            "type": "initial",
            "maximumWarning": "500kb",
            "maximumError": "1mb"
          },
          {
            "type": "anyComponentStyle",
            "maximumWarning": "2kb",
            "maximumError": "4kb"
          }
        ],
        "fileReplacements": [
          {
            "replace": "src/environments/environment.ts",
            "with": "src/environments/environment.prod.ts"
          }
        ],
        "outputHashing": "all"
      },
      "development": {
        "buildOptimizer": false,
        "optimization": false,
        "vendorChunk": true,
        "extractLicenses": false,
        "sourceMap": true,
        "namedChunks": true
      }
    },
    "defaultConfiguration": "production"
  },
  "serve": {
    "builder": "@angular-devkit/build-angular:dev-server",
    "configurations": {
      "production": {
        "browserTarget": "firstProject:build:production"
      },
      "development": {
        "browserTarget": "firstProject:build:development"
      }
    },
    "defaultConfiguration": "development"
  },
  "extract-i18n": {
    "builder": "@angular-devkit/build-angular:extract-i18n",
    "options": {
      "browserTarget": "firstProject:build"
    }
  },
  "test": {
    "builder": "@angular-devkit/build-angular:karma",
    "options": {
      "main": "src/test.ts",
      "polyfills": "src/polyfills.ts",
      "tsConfig": "tsconfig.spec.json",
      "karmaConfig": "karma.conf.js",
      "assets": [
        "src/favicon.ico",
        "src/assets"
      ]
    }
  }
}

```

```

    ],
    "styles": [
      "src/styles.css"
    ],
    "scripts": []
  }
}
}
},
"defaultProject": "OnlineCharity"
}

```

(3.) Package.json

```

{
  "name": "Online-Charity",
  "version": "0.0.0",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "watch": "ng build --watch --configuration development",
    "test": "ng test"
  },
  "private": true,
  "dependencies": {
    "@angular/animations": "~12.2.0",
    "@angular/common": "~12.2.0",
    "@angular/compiler": "~12.2.0",
    "@angular/core": "~12.2.0",
    "@angular/forms": "~12.2.0",
    "@angular/platform-browser": "~12.2.0",
    "@angular/platform-browser-dynamic": "~12.2.0",
    "@angular/router": "~12.2.0",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "mongodb": "^3.7.3",
    "rxjs": "~6.6.0",
    "tslib": "^2.3.0",
    "zone.js": "~0.11.4"
  },
  "devDependencies": {
    "@angular-devkit/build-angular": "~12.2.8",
    "@angular/cli": "~12.2.8",
    "@angular/compiler-cli": "~12.2.0",
    "@types/jasmine": "~3.8.0",
    "@types/node": "^12.11.1",
    "jasmine-core": "~3.8.0",
    "karma": "~6.3.0",
    "karma-chrome-launcher": "~3.1.0",
    "karma-coverage": "~2.0.3",
    "karma-jasmine": "~4.0.0",
    "karma-jasmine-html-reporter": "~1.7.0",
    "typescript": "~4.3.5"
  }
}

```

```
}  
}
```

(4.)Server.js

```
const http = require("http");  
const app = require("./backend/app");  
  
//const server = http.createServer((req, res)=>{  
//    res.end("Welcome to my 1st Node JS");  
//});  
  
const server = http.createServer(app);  
  
server.listen(3000, ()=>{  
    console.log("Please Check Port No. 3000");  
})
```

(5.)addCharity.component.html

```
<br><br><br><br>  
<div class="container" style="opacity: 0.8;">  
    <div style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-  
sm-8 col-sm-offset-2">  
        <div class="panel panel-info" >  
            <div class="panel-heading">  
                <div class="panel-title text-center" style="color:white;  
font-family:'time new roman'; font-size: 25px;">Start a Fundraiser</div>  
            </div>  
            <div style="padding-top:30px; background-color:#333;"  
class="panel-body" >  
                <p style="color:white;">  
                    {{msg}}  
                </p>  
                <form (submit)="doAddcharity(f)"  
#f="ngForm">  
                    <div style="margin-bottom: 25px" class="input-group">  
                        <span class="input-group-addon"><i  
class="glyphicon glyphicon-file"></i></span>  
                        <input type="file" class="form-  
control" name="bfile" ngModel>  
                    </div>  
                    <div style="margin-bottom: 25px" class="input-  
group">
```

```

        <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span>
        <input type="text" class="form-
control" name="bname" ngModel placeholder="Beneficiary Name"
>
        </div>

        <div style="margin-bottom: 25px"
class="input-group">
        <span class="input-group-addon"><i
class="glyphicon glyphicon-usd"></i></span>
        <input type="number" class="form-
control" name="bprice" ngModel placeholder="Amount to be raised"
>
        </div>

        <div style="margin-bottom: 25px"
class="input-group">
        <span class="input-group-
addon"><i class="glyphicon glyphicon-time"></i></span>
        <input type="text"
class="form-control" name="bleft" ngModel placeholder="Days Left">
        </div>

        <div style="margin-top:10px" class="form-group">
        <!-- Button -->
        <div class="col-sm-12" style="color:white;">
        <input type="submit" class="btn btn-primary
btn-lg" value="Add User"/> &nbsp;
        </div>
        </div>
        </form>
        <br> <br><br>

        </div>
    </div>
</div>

```

(6.)addCharity.component.ts

```

import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { UsersService } from '../users.service';

@Component({
  selector: 'app-addcharity',
  templateUrl: './addcharity.component.html',

```

```

    styleUrls: ['./addcharity.component.css']
  })
  export class AddcharityComponent implements OnInit {

    msg: string;

    constructor(public userSer : UsersService) { }

    ngOnInit(): void {
    }

    doAddcharity(form : NgForm)
    {
      console.log("User Added");

      console.log(form.value);

      this.userSer.addCharity(form.value).subscribe((data:string)=>{

        console.log(data);

        this.msg = data;

        form.reset();

      }, (error: any)=>{

        console.log(error);

        this.msg ="Something Went Wrong";

      });

    }

  }
}

```

(7.)addUser.component.html

```

<br><br><br><br>
<div class="container" style="opacity: 0.8;">
  <div style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-
sm-8 col-sm-offset-2">
    <div class="panel panel-info" >
      <div class="panel-heading">
        <div class="panel-title text-center" style="color:white;
font-family:'time new roman'; font-size: 25px;">Add Users</div>
      </div>
      <div style="padding-top:30px; background-color:#333;"
class="panel-body" >
        <p style="color:white;">
          {{msg}}
        </p>

```

```

        <form (submit)="addUsers(f)"
#f="ngForm">
            <div style="margin-bottom: 25px" class="input-group">
                <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span>
                <input type="text" class="form-
control" name="uname" #username="ngModel"
(blur)="usernameCheck(username.value)" ngModel placeholder="Username" required
minlength="4">
            </div>
            <p class="alert alert-danger" *ngIf =
"username.errors?.required && username.touched">
                Please Enter Username
            </p>
            <p class="alert alert-warning" *ngIf =
"username.errors?.minlength && username.touched">
                Please Enter Username minimum 6 characters
            </p>
            <div style="margin-bottom: 25px" class="input-group">
                <span class="input-group-addon"><i
class="glyphicon glyphicon-envelope"></i></span>
                <input type="text" class="form-
control" name="uemail" ngModel #useremail="ngModel" placeholder="Email"
required email>
            </div>
            <p class="alert alert-danger" *ngIf =
"useremail.errors?.required && useremail.touched">
                Please Enter Email Id
            </p>
            <p class="alert alert-danger"
*ngIf="useremail.errors?.email && useremail.touched">
                Please Enter Valid Email Id
            </p>
            <div style="margin-bottom: 25px" class="input-group">
                <span class="input-group-addon"><i
class="glyphicon glyphicon-lock"></i></span>
                <input type="password" class="form-
control" name="upassword" ngModel placeholder="Password">
            </div>
            <div style="margin-bottom: 25px" class="input-group">
                <span class="input-group-addon"><i
class="glyphicon glyphicon-phone"></i></span>
                <input type="text" class="form-
control" name="uphone" ngModel placeholder="Phone Number" pattern="^(0|[1-
9])[0-9]*$" #userphone="ngModel" maxlength="10"
minlength="10">
            </div>
            <p class="alert alert-danger"
*ngIf="userphone.errors?.pattern">

```

```

        Please Enter Valid Phone Number
    </p>

    <p class="alert alert-warning" *ngIf =
"userphone.errors?.minlength && userphone.touched">
        Invalid Phone Number!!
    </p>

    <div style="margin-top:10px" class="form-group">
        <!-- Button -->
        <div class="col-sm-12" style="color:white;">
            <input type="submit" class="btn btn-success"
value="Add User" [disabled]="f.invalid || !usernameAvail"/> &nbsp;
        </div>
    </div>
</form>
<br> <br><br>

</div>
</div>
</div>

```

(8.)addUser.component.ts

```

import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { UsersService } from '../users.service';

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

  msg : string;

  usernameAvail = false;

  constructor(public userSer : UsersService) { }

  ngOnInit(): void {
  }

  addUsers(form:NgForm)
  {
    console.log(form.value);
  }
}

```

```

    this.userSer.userRegistration(form.value).subscribe((data:string)=>{
        console.log(data);
        this.msg = data;
        form.reset();
    },(error:any)=>{
        console.log(error);
        this.msg = "Something Went Wrong";
    });
}

usernameCheck(uname : string)
{
    this.userSer.usernameAvailability(uname).subscribe((data: any[])=>{
        console.log(data);

        if(data.length==0)
        {
            this.msg = "Congrats! Username is available for you!!";
            this.usernameAvail = true;
        }
        else
        {
            this.msg = "Username is not available";
            this.usernameAvail = false;
        }
    }, (error:any)=>{
        console.log(error);
    });
}
}

```

(9.)editUser.component.html

```

<div class="container">
    <div style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-sm-8
col-sm-offset-2">
        <div class="panel panel-info" >

```

```

<div class="panel-heading">
  <div class="panel-title">Edit User</div>
</div>
<div style="padding-top:30px" class="panel-body" >
  <p>
    {{msg}}
  </p>
  <form #f="ngForm" (submit)="editUser(f)"
*ngIf="userData">
    <div style="margin-bottom: 25px" class="input-group">
      <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span>
      <input type="text" class="form-control" name="uname"
[ngModel]="userData.uname" #username="ngModel" placeholder="Username" required
minlength="4">
    </div>

    <p class="alert alert-danger"
*ngIf="username.errors?.required && username.touched">
      Please Enter Username
    </p>
    <p class="alert alert-danger"
*ngIf="username.errors?.minlength &&
username.touched">
      Please Enter Minimum 6 Characters
    </p>
    <div style="margin-bottom: 25px" class="input-group">
      <span class="input-group-addon"><i
class="glyphicon glyphicon-envelope"></i></span>
      <input type="text" class="form-control" name="uemail"
[ngModel]="userData.uemail" #useremail="ngModel" placeholder="Email" required
email>
    </div>

    <p class="alert alert-danger"
*ngIf="useremail.errors?.required && useremail.touched">
      Please Enter Your Email
    </p>
    <p class="alert alert-danger"
*ngIf="useremail.errors?.email && useremail.touched">
      Please Enter Valid Email
    </p>
    <div style="margin-bottom: 25px" class="input-group">
      <span class="input-group-addon"><i
class="glyphicon glyphicon-lock"></i></span>
      <input type="password" class="form-control" name="upassword"
[ngModel]="userData.upassword" #userpassword="ngModel" placeholder="Password"
required>
    </div>

    <div style="margin-bottom: 25px" class="input-group">
      <span class="input-group-addon"><i
class="glyphicon glyphicon-phone"></i></span>
      <input type="text" class="form-control" name="uphone"
[ngModel]="userData.uphone" placeholder="Phone Number" #userphone="ngModel"
pattern="^(0|[1-9][0-9]*)$">
    </div>

    <p class="alert alert-danger"
*ngIf="userphone.errors?.pattern">
      Please Enter Nos Only
    </p>

```

```

        <div style="margin-top:10px" class="form-group">
            <!-- Button -->
            <div class="col-sm-12">
                <input type="submit" class="btn btn-success" value="Update"
[disabled]="f.invalid" /> &nbsp;
            </div>
        </div>
    </form>
    <br> <br><br>

</div>
</div>
</div>

```

(10.)editUser.component.ts

```

import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { ActivatedRoute, Params } from '@angular/router';
import { UsersService } from '../users.service';

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

  userData : { _id : number, uname: string, upassword : string, uemail : string, uphone : string };

  msg : string;

  constructor(public activeRoute : ActivatedRoute, public userSer : UsersService) { }

  ngOnInit(): void {

    this.activeRoute.params.subscribe((param:Params)=>{

      console.log(param);

      if(param.userid)
      {
        this.userSer.getSingleUserData(param.userid).subscribe((data:any[])=>{

          console.log(data);

          this.userData = data[0];

        }, (error:any)=>{

          console.log(error);

        });
      }
    });
  }
}

```

```

    });
  }

  editUser(form:NgForm)
  {
    form.value._id = this.userData._id;

    this.userSer.editSingleUserData(form.value).subscribe((data:string)=>{

      console.log(data);

      this.msg = data;

    }, (error:any)=>{

      console.log(error);

      this.msg = "Something Went Wrong";
    });
  }
}

```

(11.)login.component.html

```

<br><br><br><br>
<div class="container" style="opacity: 0.8;">
  <div style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-
sm-8 col-sm-offset-2">
    <div class="panel panel-info" >
      <div class="panel-heading">
        <div class="panel-title text-center" style="color:white;
font-family:'time new roman'; font-size: 25px;">Log In</div>
      </div>
      <div style="padding-top:30px; background-color:#333;"
class="panel-body" >
        <p style="color:#FF2451;">
          {{msg}}
        </p>
        <form (submit)="doLogin(f)"
#f="ngForm">
          <div style="margin-bottom: 25px" class="input-group">
            <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span>
            <input type="text" class="form-
control" name="uname" ngModel placeholder="username"
>
          </div>

          <div style="margin-bottom: 25px" class="input-group">
            <span class="input-group-addon"><i
class="glyphicon glyphicon-lock"></i></span>
            <input type="password" class="form-
control" name="upassword" ngModel placeholder="password">

```



```

    {
      localStorage.setItem("loggeduser",data[0]._id);
      this.myRouter.navigateByUrl("/charity");
    }

  },(error: any)=>{

    console.log(error);
    this.msg = "Something Went Wrong";
  });
}
}

```

(13.)register.component.html

```

<br><br><br><br>
<div class="container" style="opacity: 0.8;">
  <div style="margin-top:50px;" class="mainbox col-md-6 col-md-offset-3 col-
sm-8 col-sm-offset-2">
    <div class="panel panel-info" >
      <div class="panel-heading">
        <div class="panel-title text-center" style="color:white;
font-family:'time new roman'; font-size: 25px;">Register</div>
      </div>
      <div style="padding-top:30px; background-color:#333;"
class="panel-body" >
        <p style="color:white;">
          {{msg}}
        </p>
        <form (submit)="doRegistration(f)"
#f="ngForm">
          <div style="margin-bottom: 25px" class="input-group">
            <span class="input-group-addon"><i
class="glyphicon glyphicon-user"></i></span>
            <input type="text" class="form-
control" name="uname" #username="ngModel" ngModel placeholder="Username"
required minlength="4">
          </div>
          <p class="alert alert-danger" *ngIf =
"username.errors?.required && username.touched">
            Please Enter Username
          </p>
          <p class="alert alert-warning" *ngIf =
"username.errors?.minlength && username.touched">
            Please Enter Username minimum 6 characters
          </p>
          <div style="margin-bottom: 25px" class="input-group">
            <span class="input-group-addon"><i
class="glyphicon glyphicon-envelope"></i></span>

```

```

        <input type="text" class="form-
control" name="uemail" ngModel #useremail="ngModel" placeholder="Email"
required email>
    </div>

    <p class="alert alert-danger" *ngIf =
"useremail.errors?.required && useremail.touched">
        Please Enter Email Id
    </p>

    <p class="alert alert-danger"
*ngIf="useremail.errors?.email && useremail.touched">
        Please Enter Valid Email Id
    </p>
    <div style="margin-bottom: 25px" class="input-group">
        <span class="input-group-addon"><i
class="glyphicon glyphicon-lock"></i></span>
        <input type="password" class="form-
control" name="upassword" ngModel placeholder="Password">
    </div>

    <div style="margin-bottom: 25px" class="input-group">
        <span class="input-group-addon"><i
class="glyphicon glyphicon-phone"></i></span>
        <input type="text" class="form-
control" name="uphone" ngModel placeholder="Phone Number" pattern="^(0|[1-
9][0-9]*)$" #userphone="ngModel" maxlength="10"
minlength="10">
    </div>

    <p class="alert alert-danger"
*ngIf="userphone.errors?.pattern">
        Please Enter Valid Phone Number
    </p>

    <p class="alert alert-warning" *ngIf =
"userphone.errors?.minlength && userphone.touched">
        Invalid Phone Number!!
    </p>

    <div style="margin-top:10px" class="form-group">
        <!-- Button -->
        <div class="col-sm-12" style="color:white;">
            <input type="submit" class="btn btn-success"
value="Register" [disabled]="f.invalid"/> &nbsp;
            Already have Account? <a routerLink="/login"
style="color:#FF2451;">Login</a>
        </div>
    </div>
</form>
<br> <br><br>

</div>
</div>
</div>

```

</div>

(14.)register.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { UsersService } from '../users.service';

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

  msg : string;

  constructor(public userSer : UsersService) { }

  ngOnInit(): void {
  }

  doRegistration(form: NgForm)
  {
    console.log("User Registered");

    console.log(form.value);

    this.userSer.userRegistration(form.value).subscribe((data:string)=>{

      console.log(data);

      this.msg = data;

      form.reset();

    },(error:any)=>{

      console.log(error);

      this.msg ="Something Went Wrong";

    });

  }

}
```

(15.)users.component.html

```

<div class="container">
  <div class="row" style="padding-top: 30px;">
    <div class="col-sm-4"></div>

    <div class="col-sm-4"></div>

    <div class="col-sm-4" style="color:white; font-family:'Times New Roman';
font-size: 16px;">
      Serach: <input type="text" #search (keyup)="doSearch(search.value)"
style="border-radius:5px; padding: 5px;"> &nbsp;
      <a routerLink="/adduser"><button class="btn btn-success">Add
User</button></a>
    </div>
  </div>
<br><br>
  <div class="row">
<div class="col"></div>

<div class="col">
  {{msg}}
  <table class="table table-striped">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Name</th>
        <th scope="col">Email</th>
        <th scope="col">Phone</th>
        <th scope="col">Operations</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let user of users; index as i">
        <th scope="row">{{i+1}}</th>
        <td>{{user.username}}</td>
        <td>{{user.uemail}}</td>
        <td>{{user.uphone}}</td>
        <td>
          <a routerLink="/edituser/{{user._id}}"> <button class="btn btn-
primary">Edit</button></a> &nbsp;
          <button class="btn btn-danger"
(click)="deleteUser(user._id)">Delete</button>
        </td>
      </tr>
    </tbody>
  </table>
</div>

<div class="col"></div>
</div>
</div>
<br><br><br><br>

```

(16.)users.component.ts

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

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

  users : any[] = [];

  msg : string;

  constructor(public userSer : UsersService) { }

  ngOnInit(): void {

    this.userSer.getAllUsers().subscribe((data: any[])=>{

      console.log(data);
      this.users = data;

    },(error:any)=>{

      console.log(error);
      this.msg = "Something Went Wrong";
    })

  }

  deleteUser(userId:number)
  {
    if(confirm("Are you sure want to Delete this User"))
    {
      //console.log("User Deleted with user id "+userId);

      this.userSer.deleteUserData(userId).subscribe((data:string)=>{

        this.msg=data;

        var index = this.users.findIndex((obj)=>{

          return obj._id == userId;
        });
        this.users.splice(index,1);
      }, (error:any)=>{

        console.log(error);
        this.msg = "Something Went Wrong";

      }));
    }
  }

  doSearch(search:string)
  {
    this.userSer.searchUsers(search).subscribe((data:any[])=>{
```

```

    console.log(data);
    this.users = data;

  }, (error:any)=>{

    console.log(error);

  });
}
}

```

OUTPUT SNAPSHOT:

The screenshot shows the MongoDB Atlas web interface. The left sidebar displays the 'DEPLOYMENT' section with 'Databases' expanded, showing a tree structure with 'collegebatch1' and its sub-databases 'charity' and 'users'. The main panel is titled 'Find' and shows 'QUERY RESULTS 1-4 OF 4'. The results are as follows:

Document
{ "_id": "1636659803199", "mfile": "C:\\fakepath\\Save Children.jpg", "mname": "Save Children", "mprice": 18000, "malbum": "500" }
{ "_id": "1636660259055", "mfile": "C:\\fakepath\\COVID Relief Fund.png", "mname": "COVID Relief fund", "mprice": 60000, "malbum": "124" }
{ "_id": "1636800713103", "mfile": "C:\\fakepath\\Cancer Care.jpg", "mname": "Cancer Care", "mprice": 500, "malbum": "12" }
{ "_id": "1636832546848", "mfile": "C:\\fakepath\\Cancer Care.jpg", "mname": "Cancer Care", "mprice": 500, "malbum": "12" }

The screenshot shows the MongoDB Atlas web interface. The left sidebar displays the 'DEPLOYMENT' section with 'Databases' expanded, showing a tree structure with 'collegebatch1' and its sub-databases 'charity' and 'users'. The main panel is titled 'Find' and shows 'QUERY RESULTS 1-1 OF 1'. The results are as follows:

Document
{ "_id": "1636832546848", "username": "saurya", "email": "sauryakushta100@gmail.com", "password": "1234", "uphone": "9264453913" }

```
Windows PowerShell
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Asus\Desktop\firstProject>ng serve -o
✓ Browser application bundle generation complete.

Initial Chunk Files | Names          | Size
vendor.js           | vendor         | 2.80 MB
styles.css, styles.js | styles        | 519.76 kB
polyfills.js        | polyfills     | 510.58 kB
main.js             | main          | 154.04 kB
runtime.js          | runtime       | 6.63 kB

| Initial Total | 3.96 MB

Build at: 2021-11-13T19:45:02.127Z - Hash: 201ca2acfdcb1a28a536 - Time: 12000ms

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

✓ Compiled successfully.
```