

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 Technology Degree in
Information Technology

By

AARTHI N
REG. NO. 39120002



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

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

(Established under section 3 of UGC Act,1956)

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

www.sathyabamauniversity.ac.in

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 **AARTHI.N** (Reg. No: 39120002) who carried out the project entitled "**Student Infotainment App Using Angular JS**" under my supervision from September 2021 to November 2021.

Internal Guide

Dr. G MATHIVANAN, M.E., Ph.D.,

Head of the Department

Dr. R SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **AARTHI.N** hereby declare that the project report entitled “**Student Infotainment App Using Angular JS**” done by me under the guidance of **Dr. G. MATHIVANAN** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Information Technology.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **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** and **Dr. R. Subhashini, M.E., Ph.D. Head of the Department of Information Technology** for providing me 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. G. MATHIVANAN M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement 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 the project.

TRAINING CERTIFICATE



ABSTRACT

- ✓ Angular is a platform and framework for building single-page client applications using HTML and TypeScript.
- ✓ It implements core and optional functionality as a set of TypeScript libraries that we import into our applications.
- ✓ The architecture of an Angular application relies on certain fundamental concepts. The basic building blocks are the NgModules.
- ✓ In this project many components are created and routes are been set up between each components for the dynamic motion of one page to another page.
- ✓ In this Student Infotainment App, the staff who has account can login , the staffs who are not having the account can register themselves and they can login.
- ✓ Once after they login, they can view the student details and the registered staffs/admin can add, edit, search and delete the registered players' information via the users component.
- ✓ For the backend purpose, the app is connected to the mongodb so that the information of the students will be displayed in the mongo database.
- ✓The admin action will not only carried out in front end, but also in backend in mongodb

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
	ABSTRACT	I
	LIST OF FIGURES	li
	LIST OF TABLES	lii
	LIST OF ABBREVIATIONS	iv
1	INTRODUCTION	1
	1.1 WHAT IS ANGULAR JS?	1
	1.2 ROLE OF HTML, CSS & JS IN ANGULAR	2
	1.3 REQUIREMENTS FOR ANGULAR PROJECT	3
2.	STUDENT INFOTAINMENT APP	
	2.1 OVERVIEW	4
	2.2 FUNCTIONALREQUIREMENTS	4
	2.3 COMPONENTS	4
	2.4 OVERVIEW OF COMPONENTS	5
	2.5 BACKEND	17
3.	SUMMARY AND CONCLUSIONS	18
	REFERENCE	18
	APPENDIX	19
	A.SOURCE CODE	19
	B.SCREENSHOTS	44

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
1.1	Home Page	5
1.2	About us Page	6
1.3	Courses Offered Page	7
1.4	Contact Us Page	8
1.5	Login Page	9
1.6	Register Page	10
1.7	Student Details Page	11
1.7.1	Add Student Details Page	12
1.7.2	Edit Student Details Page	13
1.7.3	Search Student Details Page	14
1.7.4	Delete Student Details Page	15
1.8	Logout Page	16
1.9	MongoDB connection	17

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
2.1	Login Fields	9
2.2	Register/Sign-up Fields	10
2.3	Add Student Fields	12
2.4	Edit Student Fields	13

LIST OF ABBREVIATIONS

ABBREVIATION		EXPANSION
Mongodb	-	Mongo database
App	-	Application
Chars	-	Characters
No's	-	Numbers
Acc	-	According
JS	-	Java Script

1).INTRODUCTION

1.1What is Angular Js?

❖ The AngularJS 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.

❖ AngularJS 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 AngularJS 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.

1.2 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

1.3 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.

2) STUDENT INFOTAINMENT APP

- This project aims to maintain the information about the students in the universities/colleges. It will help the univ to manage the details of all the students.

- Staff will be able to register themselves and search the data of the students using search page. They are allowed to update the students information.

- The Student Infotainment App designed by Team Amigas will be very easy and a staffs can view the student details at any time . Now a days, internet is the most widely used network in modern world and it's quite a tough job to take all the records and bundle of files to view the student details . For their easy purpose, Team Amigas made this Student Infotainment App, so that staffs easily view the student details anytime , anywhere.

2.2 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

2.3 COMPONENTS:

- ❖ Home
- ❖ About Us
- ❖ Courses Offered
- ❖ Contact Us
- ❖ Login
- ❖ Register/Signup
- ❖ Student Details Management(for admin purpose)
 - Add Student
 - Edit Student
 - Search Student
 - Delete Student
- ❖ Logout

2.4 OVERVIEW OF COMPONENTS:

- HOME

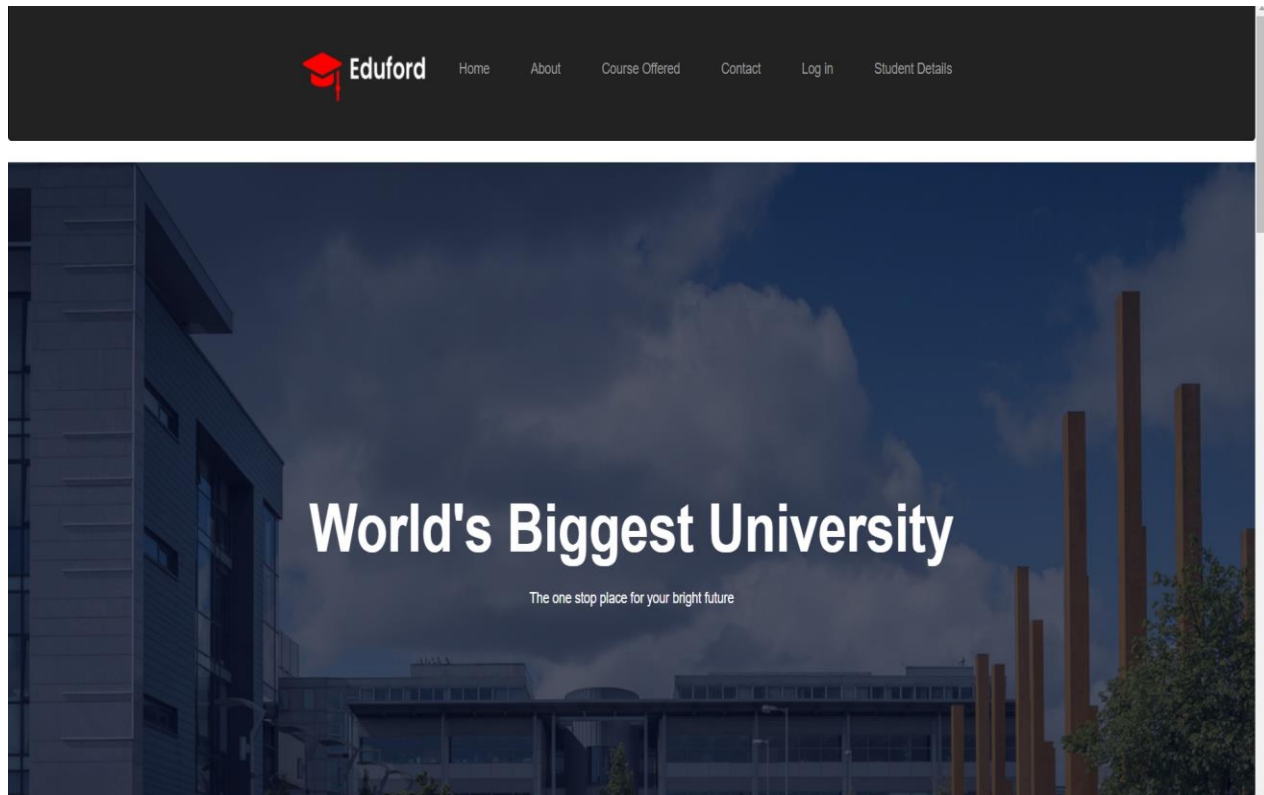


Figure 1.1 Home page

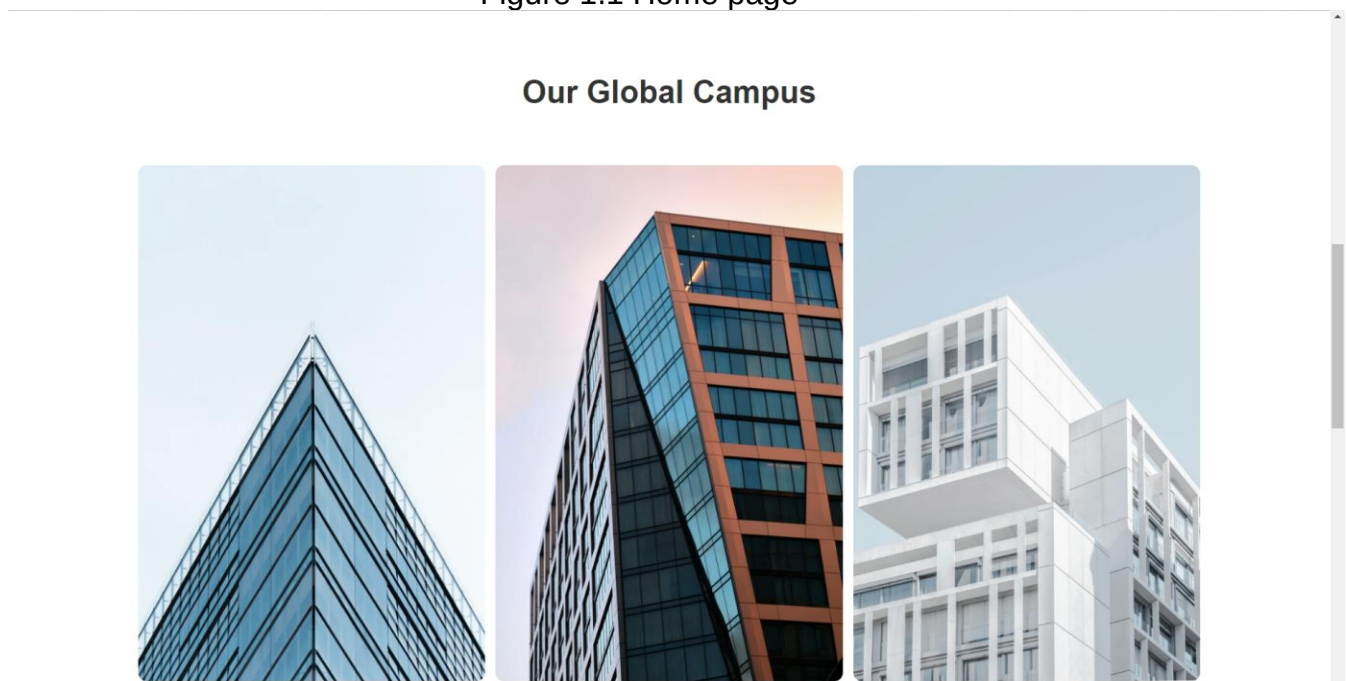


Figure 1.1.1 Home Page
(5)

Our Facilities

We commit to meet the financial need of every admitted undergraduate student. We don't consider your ability to pay when we review your application, and we don't base our decision on whether you can cover the cost. If you're accepted here, you belong here./p>



World Class Library

The Central Library is the indispensable hub of discovery, learning and creativity at DEI by providing a distinguished collection of information resources unbounded by place and preserved for future generations.



Largest Play Ground

A playground, playpark, or play area is a place designed to provide an environment for students that facilitates play, typically outdoors and indoors.



Tasty and Healthy Food

Canteens and cafeterias are one of the important parts of student life where you can find most of students chatting and gossiping on different topics. It plays a vital role in removing tensions and refreshing students. The college canteen is fairly new and provides enough and delicious food at affordable rates

Figure 1.1.2 Home Page

• ABOUT US



[Home](#)

[About](#)

[Course Offered](#)

[Contact](#)

[Log in](#)

[Student Details](#)

We are the World's Largest University

we're a leading research university with a heart.Founded in the decade that the U.S Constitution was signed , we're the nations oldest Catholic and Jesuit University.Today we're a forward-looking, diverse community devoted to social justice, restless inquiry and respect for each person's individual needs and talents.



Figure 1.2 About Us Page
(6)

- **Courses Offered**

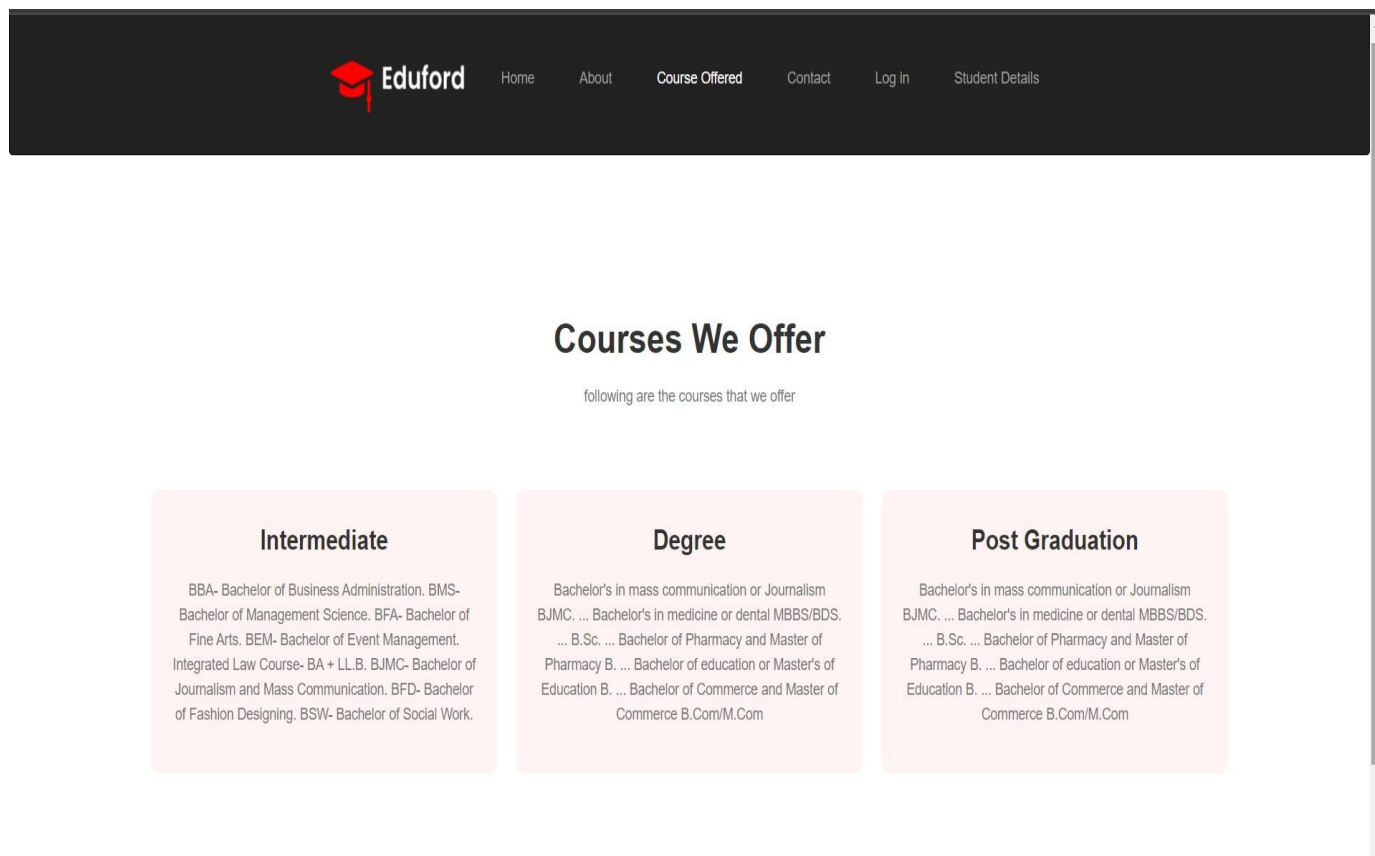


Figure 1.3 Courses Offered Page

• Contact Us Page

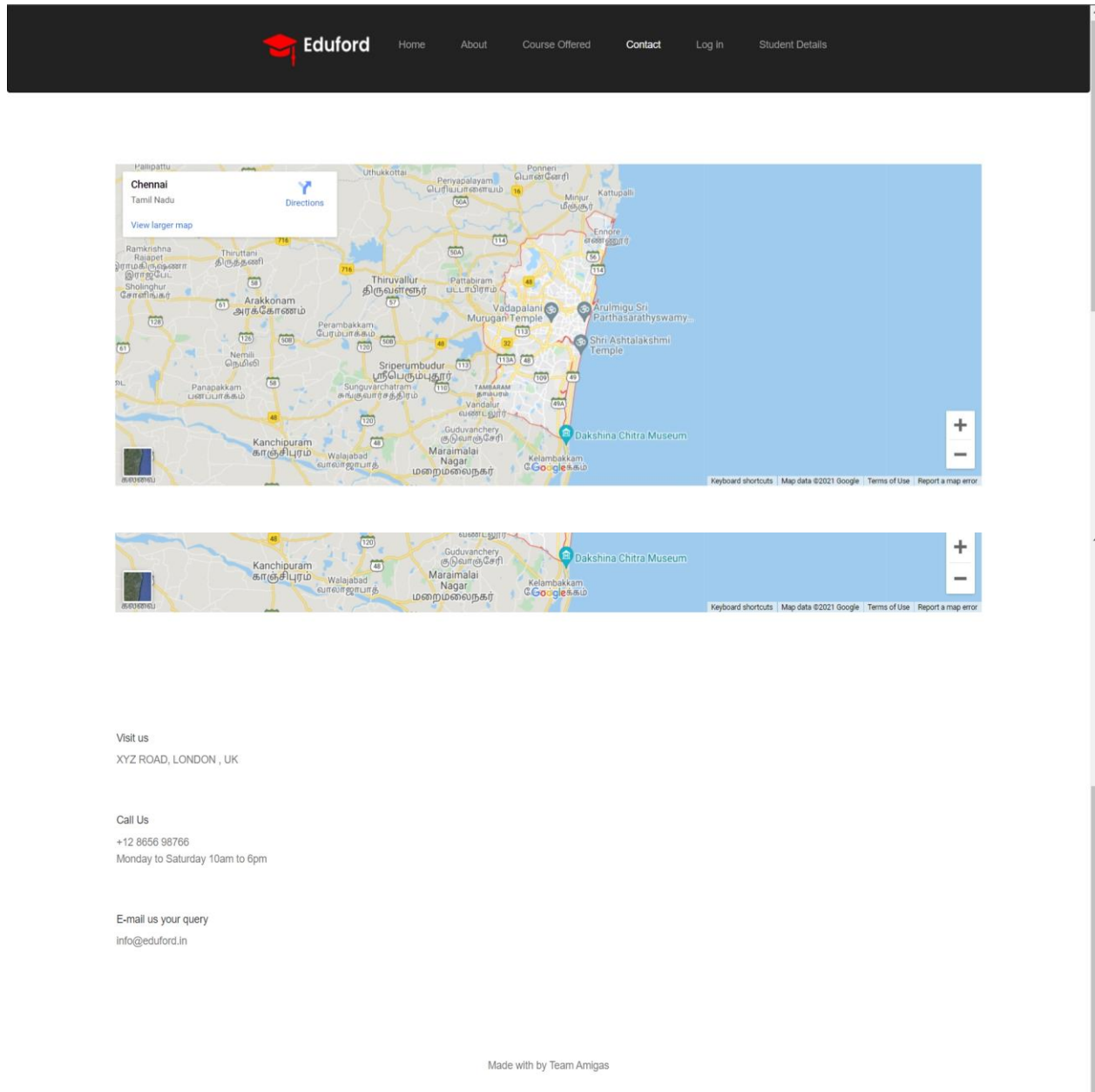


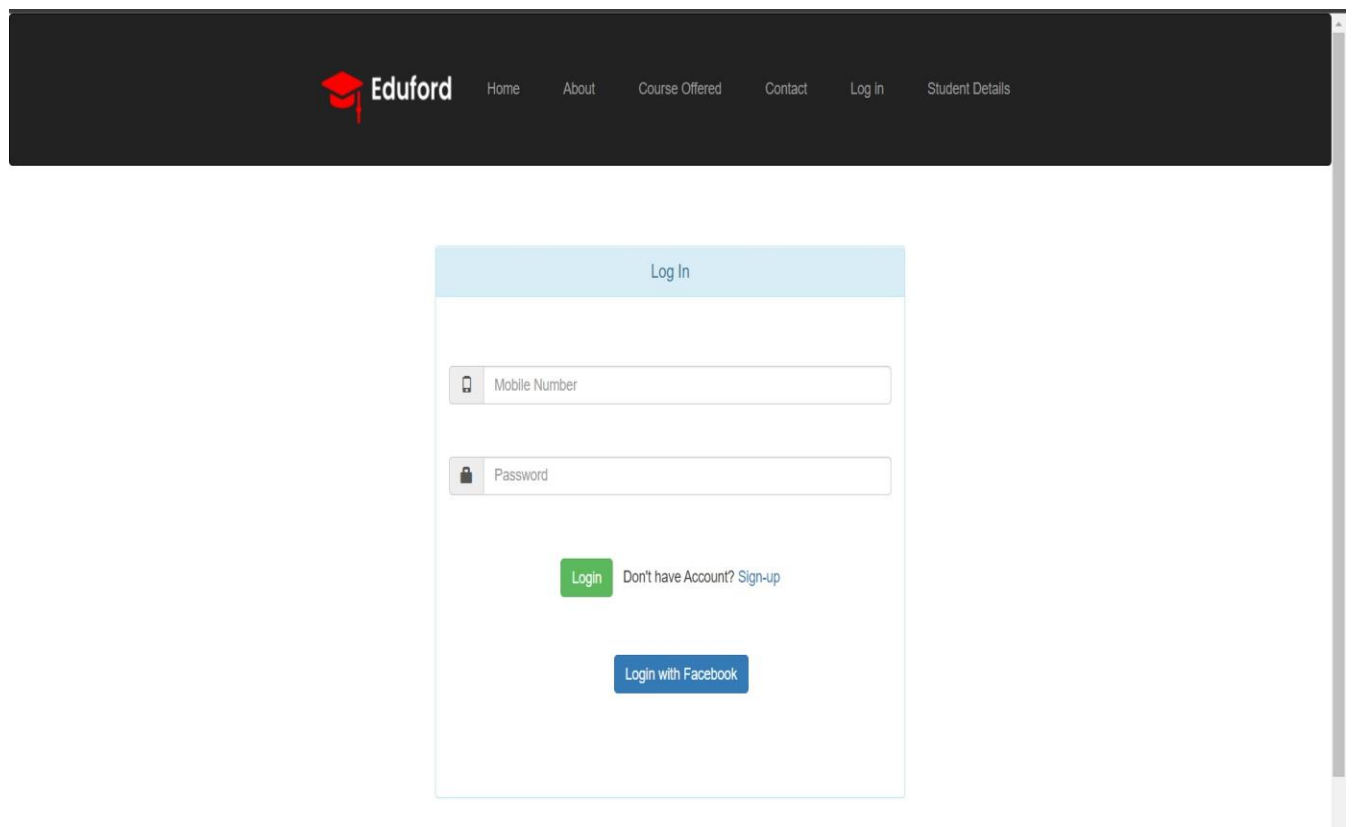
FIGURE 1.4 Contact Us Page

- **Login portal:**

The login portal is used for the registered staffs to login into the app. The admin also can login to do the student details related activities.

Fields	Validations
Email	Required Email validation
Password	Required

Table 2.1 Login Fields



The screenshot shows the Eduford login interface. At the top is a dark navigation bar with the Eduford logo and links for Home, About, Course Offered, Contact, Log in, and Student Details. The main login form is centered and titled 'Log In'. It includes input fields for 'Mobile Number' and 'Password'. Below these fields are three options for logging in: a green 'Login' button, a link to 'Sign-up' for users without an account, and a blue 'Login with Facebook' button.

Fig.1.5 Login Page

- **Register/Sign-up portal:**

User will be able to register themselves by clicking sign-up link from login screen. On submission, data gets inserted to the database via mongodb.

FIELDS	VALIDATION
Username	Required, Minlength = 5
E-mail	Required, E-mail validation
Password	Minlength = 6
Re-type Password	Same as password
Phone no	Number validation, Minlength = 6

Table 2.2: Register / Sign-up fields

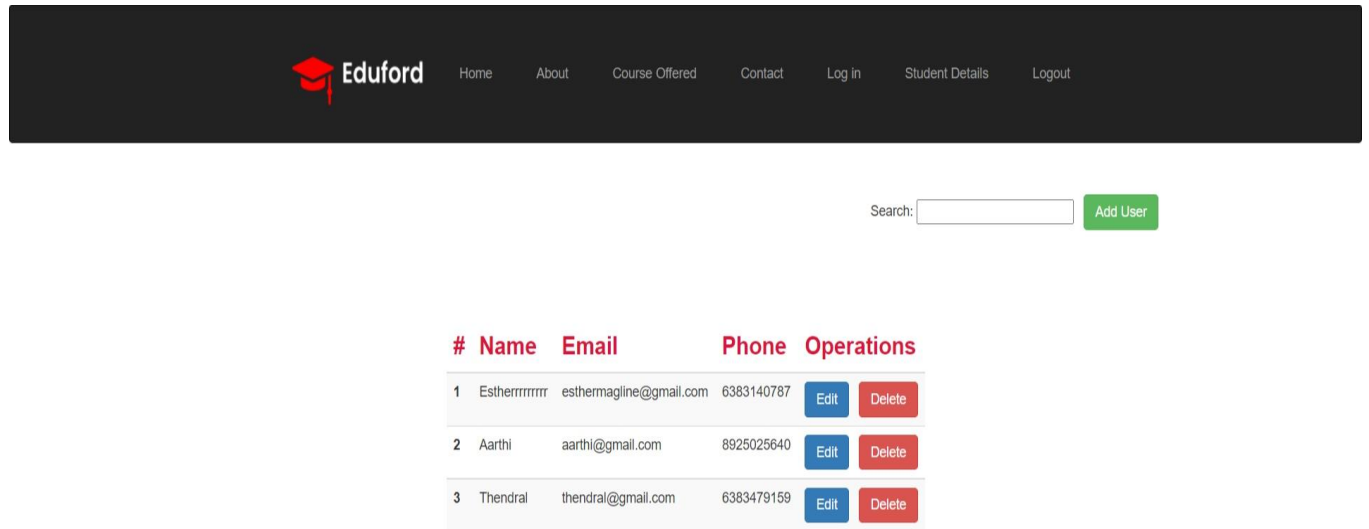
The screenshot shows the Eduford website header with navigation links: Home, About, Course Offered, Contact, Log in, and Student Details. Below the header is a 'Staff Sign Up' form with the following fields: Username, Email, Password, Re enter password, and Phone Number. A green 'Register' button is located below the Phone Number field, followed by the text 'Already have Account? Login' with a blue link to 'Login'.

Made with by Team Amigas
Figure 1.6 Signup Page

- **Student Details**

This portal is fully for the purpose of admin in-order to add user, edit information of already registered player and delete the user from the management portal.

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



#	Name	Email	Phone	Operations
1	Estherrrrrrr	esthermagline@gmail.com	6383140787	Edit Delete
2	Aarthi	aarthi@gmail.com	8925025640	Edit Delete
3	Thendral	thendral@gmail.com	6383479159	Edit Delete

Figure 1.7 Student Details

- **Add Student:**

Admin user will have privileges to add a new student by clicking the Add Student link.

FIELDS	VALIDATION
Username	Required, Minlength = 6
E-mail	Required, E-mail validation
Password	Required, Minlength = 6
Ph.no	Pattern validation, Minlength = 6

Table 2.3 Add Student Fields

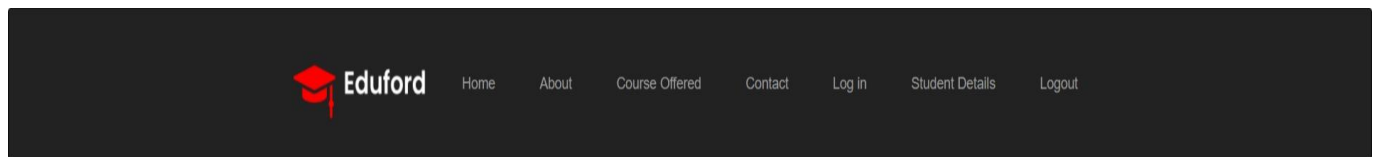


Figure 1.7.1 Add user portal

▪ Edit User Portal:

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

FIELDS	VALIDATION
Username	Username gets auto-updated
E-mail	E-mail gets auto-updated
Password	Password gets auto-updated
Ph.no	Phone No gets auto-updated

Table 2.4 Edit User Portal Fields

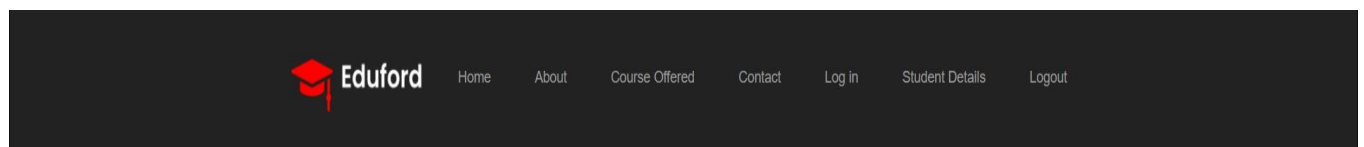
A screenshot of the "Edit User" form. The form has a light blue header with the text "Edit User". Below the header, there are four input fields, each with a small icon on the left: a person icon for the username field containing "Estherrrrrrr", an envelope icon for the email field containing "esthermagline@gmail.com", a lock icon for the password field containing "*****", and a phone icon for the phone number field containing "6383140787". At the bottom of the form is a green button with the text "Update" in white.

Figure 1.7.2 Edit User Portal

▪ Search Student Option :

In Student details portal, there exists a dialogue box which does the search operations. In that search box, admin can search the student by entering their name and the info of the person who belongs to that username entered gets filtered out from the existing list of students. And another feature is when you entered the student name who doesn't register in the registration portal, the database shows that no records found.

The screenshot shows the Eduford web application interface. At the top is a dark navigation bar with the Eduford logo and links for Home, About, Course Offered, Contact, Log in, Student Details, and Logout. Below the navigation bar is a search area with a text input field containing 'es' and a green 'Add User' button. Underneath the search bar is a table with the following data:

#	Name	Email	Phone	Operations
1	Estherrrrrrr	esthermagline@gmail.com	6383140787	Edit Delete

The second screenshot shows the same Eduford web application interface. The search input field now contains 'mxd' and the 'Add User' button is still present. Below the search bar, the table headers are visible, but the content area displays 'No Records Found'.

Figure 1.7.3 Search User Portal

▪ Delete Student

Same like edit option, there exists a delete button in side of every student'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.

The screenshot shows a web browser window with the URL `localhost:4200/users`. A modal dialog box is open, titled "localhost:4200 says", with the message "Are you sure want to delete this users?". It has "OK" and "Cancel" buttons. Below the dialog, there is a search bar and an "Add User" button. A table lists four users with columns for #, Name, Email, Phone, and Operations. The "Operations" column contains "Edit" and "Delete" buttons for each user. Below the table, a message "User Deleted Successfully" is displayed. The table data is as follows:

#	Name	Email	Phone	Operations
1	Estherrrrrrr	esthermagline@gmail.com	6383140787	<button>Edit</button> <button>Delete</button>
2	Aarthi	aarthi@gmail.com	8925025640	<button>Edit</button> <button>Delete</button>
3	Thendral	thendral@gmail.com	6383479159	<button>Edit</button> <button>Delete</button>
4	nancymo	nancy@gmail.com	1234567890	<button>Edit</button> <button>Delete</button>

Below the table, a message "User Deleted Successfully" is displayed. The table data is as follows:

#	Name	Email	Phone	Operations
1	Estherrrrrrr	esthermagline@gmail.com	6383140787	<button>Edit</button> <button>Delete</button>
2	Aarthi	aarthi@gmail.com	8925025640	<button>Edit</button> <button>Delete</button>
3	Thendral	thendral@gmail.com	6383479159	<button>Edit</button> <button>Delete</button>

Figure 1.7.4 Delete User Portal

- **Logout:**

After completing all the operations. The admin/staff can logout form the portal and after clicking logout button, the user/player will be directed again to login page which means if the player has to login again if he wants to login. And this is the final stage of operating process!!

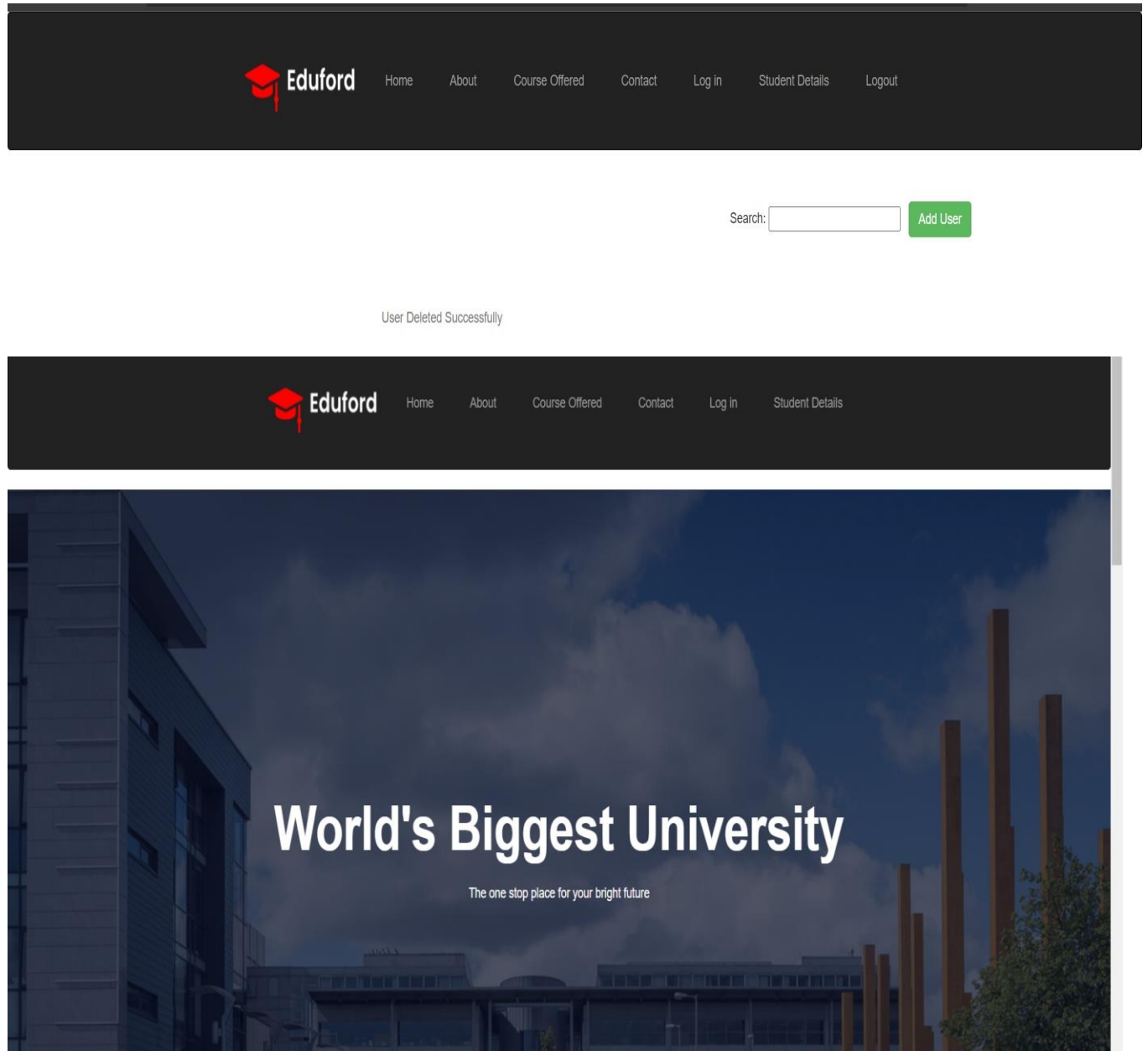


Figure 1.8 Logout Portal

BACKEND:

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

For this to be carried out, first of all a mongodb cluster should be created and a database should be made and thereby it should be connected with backend through the mongodb 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 response of the server.

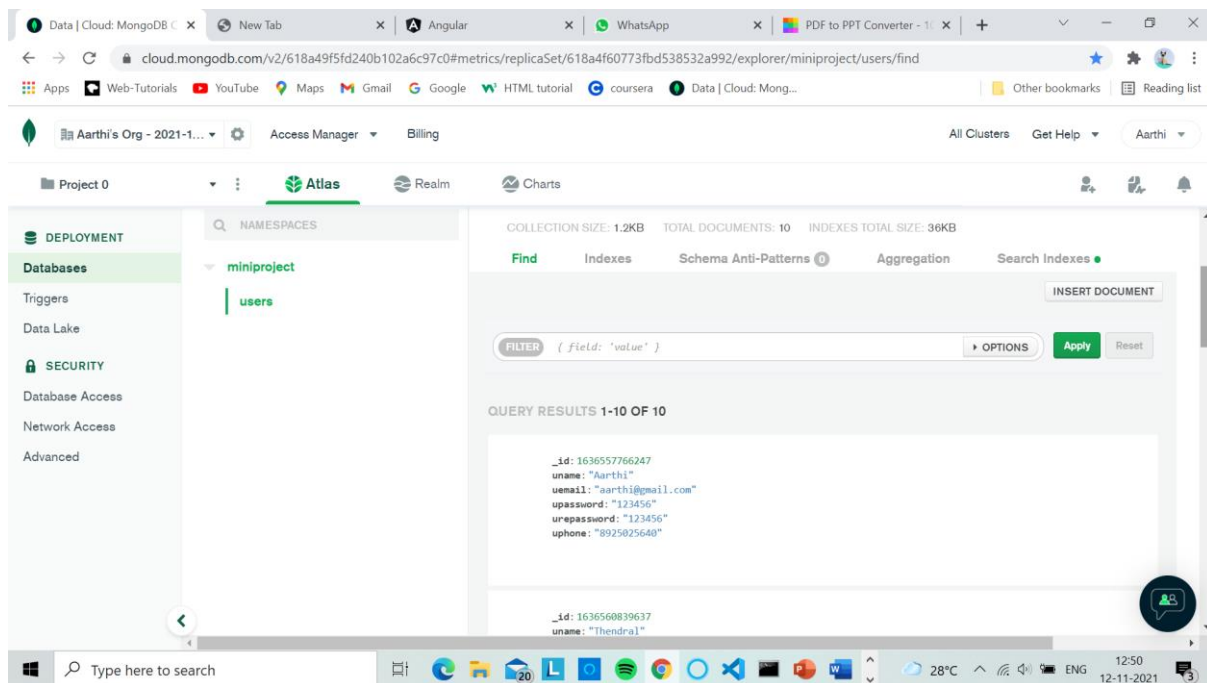


Figure 1.9 MongoDB connection

SUMMARY AND CONCLUSIONS

• SUMMARY:

This Student Infotainment app helps the staffs/admin/University to maintain the student details from all the department in a single page.

This app is a single page web application build with Angular JS. More technical and database based activities are carried out to build this app. The code is done from the scratch and the layout is made. The data is passed through database and collected in mongodb. This will be more helpful in maintaining the records!!!

• CONCLUSION:

With the steps shown in the above section, we can make lot of single page applications that helps in the modern technology. Angular JS makes the building of angular application easier as compare to many application languages. Working with angular gives more knowledge in importing modules and dealing with routes. It also involves processes in backend and the connection between the application and mongodb database also exists. Angular is now an emerging technology which helps the full stack developers in developing their own application.

REFERENCE :

1) G. Prabagaren, "Systematic approach for validating Java-MongoDB Schema," International Conference on Information Communication and Embedded Systems (ICICES2014), 2014, pp. 1-4, doi: 10.1109/ICICES.2014.7033768 and this website <https://amalendu315.github.io/Edu-tech-theme/> from chrome has been taken as an reference

APPENDIX

A.SOURCE CODE

• 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://esther:esther@mycluster.hvlzy.mongodb.net/myFirstDatabase?retryWrites=true&w=majority", (error, result)=>{

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

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

    console.log("Middleware 1");
    next();
});

app.use("/home", (req, res, next)=>{                        // middleware only for the /home 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</h1>");
});

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

    console.log("Home Page");

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

    res.json(data);
});

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

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

    console.log(req.body);

    db.collection("users").insertOne(req.body, (error, data)=>{

        if(error)
        {
            res.status(403).json("Error in Inserting Doc");
        }
        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 Doc");
        }
        else {

```

```

        res.json(data);
    }

});

});

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

    db.collection("users").find(null, {projection: {upassword : 0,urepassword
: 0}}).toArray((error, data)=>{

        if(error)
        {
            res.status(403).json("Error in Finding the Doc");
        }
        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 the Doc");
        }
        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 Doc");
        }
        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 Doc");
        }
        else {
            res.json("User Data 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("/searchuser/:searchtxt?", (req, res)=>{

    console.log(req.params.searchtxt);

    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);
    });
});

module.exports = app;

```

• HEADER COMPONENT

HTML:

```

<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand">
        
      </a>
    </div>
    <ul class="nav navbar-nav navbar-right">
      <li style="color: white; padding: 10px;"><a href="#">Home</a>
      <li style="color: white; padding: 10px;"><a routerLink="about">About <
/a></li>
      <li style="color: white; padding: 10px;"><a routerLink="/course">Course
Offered</a></li>

```

```

        <li style="color: white; padding: 10px;"><a routerLink="/contact">Contact</a></li>

        <li style="color: white; padding: 10px;"><a routerLink="/login">Log in</a></li>
        <li style="color: white; padding: 10px;"><a routerLink="/users">Student Details</a></li>

        <li style="color: white; padding: 10px;"
            *ngIf="userSer.isLoggedIn()">
            <a (click)="doLogout()">Logout</a>
        </li>
    </ul>
</div>

</nav>

```

• HOME COMPONENT

HTML

```

<section class="header">
    <div class="nav-links" id="navLinks">
        <div class="text-box">
            <h1>World's Biggest University</h1>
            <p>The one stop place for your bright future</p>
        </div>
    </div>
</section>
<section class="campus">
<h1>Our Global Campus</h1>

<div class="row">
<div class="campus-col">
    
    <div class="layer">
        <h3>LONDON</h3>
    </div>
</div>
<div class="campus-col">
    
    <div class="layer">
        <h3>NEWYORK</h3>
    </div>
</div>
<div class="campus-col">
    
    <div class="layer">
        <h3>WASHINGTON</h3>
    </div>
</div>

```

```

    </div>
</div>
</div>
</section>
<section class="facilities">
<h1>Our Facilities</h1>
<p>We commit to meet the financial need of every admitted undergraduate student. We don't consider your ability to pay when we review your application, and we don't base our decision on whether you can cover the cost. If you're accepted here, you belong here.</p>

<div class="row">
<div class="facilities-col">
    
    <h3>World Class Library</h3>
    <p>The Central Library is the indispensable hub of discovery, learning and creativity at DEI by providing a
        distinguished collection of information resources unbounded by place and preserved for future generations.</p>
</div>
<div class="facilities-col">
    
    <h3>Largest Play Ground</h3>
    <p>A playground, playpark, or play area is a place designed to provide an environment for students that facilitates play, typically outdoors and indoors. </p>
</div>
<div class="facilities-col">
    
    <h3>Tasty and Healthy Food</h3>
    <p>Canteens and cafeterias are one of the important parts of student life where you can find most of students chatting and gossiping on different topics.
        It plays a vital role in removing tensions and refreshing students.
        The college canteen is fairly new and provides enough and delicious food at affordable rates. </p>
</div>
</div>
</div>
</section>

```

CSS

```

.campus{
    width: 80%;
    margin: auto;
    text-align: center;
    padding-top: 50px;
}

```

```
.campus-col{
  flex-basis: 32%;
  border-radius: 10px;
  margin-bottom: 30px;
  position: relative;
  overflow: hidden;
}
.campus-col img{
  width: 100%;
  display: block;
}
.layer{
  background: transparent;
  height: 100%;
  width: 100%;
  position: absolute;
  top: 0;
  left: 0;
  transition: 0.5s;
}
.layer:hover{
  background: rgba(226,0,0,0.7);
}
.layer h3{
  width: 100%;
  font-weight: 500;
  color: #fff;
  font-size: 26px;
  bottom: 0;
  left: 50%;
  transform: translateX(-50%);
  position: absolute;
  transition: 0.5s;
  opacity: 0;
}
.layer:hover h3{
  bottom: 49%;
  opacity: 1;
}
.facilities{
  width: 80%;
  margin: auto;
  text-align: center;
  padding-top: 100px;
}
.facilities-col{
  flex-basis: 31%;
  border-radius: 10px;
```

```

        margin-bottom: 5%;
        text-align: left;
    }
    .facilities-col img{
        width: 100%;
        border-radius: 10px;
    }
    .facilities-col p{
        padding: 0;
    }
    .facilities-col h3{
        margin-top: 16px;
        margin-bottom: 15px;
        text-align: left;
    }
}

```

• ABOUT COMPONENT

HTML

```

<section class="about-us">
  <div class="row">
    <div class="about-col">
      <h1>We are the World's Largest University</h1>
      <p>
        we're a leading research university with a heart.Founded in the
        decade that the U.S Constitution was signed , we're the nations oldest Cathol
        ic and Jesuit University.Today we're a fprward-
        looking, diverse community devoted to social justice, restless inquiry and res
        pect for each person's individual needs and talents.
      </p>
    </div>
    <div class="about-col">
      
    </div>
  </div>
</section>

```

CSS

```

.about-us{
  width: 80%;
  margin: auto;
  padding-top: 80px;
  padding-bottom: 50px;
}

.about-col{
  flex-basis: 48%;
  padding: 30px 2px;
}

```

```

}

.about-col img{
    width: 100%;
}

.about-col h1{
    padding-top: 0;
}

.about-col p{
    padding: 15px 0 25px;
}

```

• COURSE COMPONENT

HTML

```

<section class="course">
    <h1>Courses We Offer</h1>
    <p>following are the courses that we offer</p>

    <div class="row">
        <div class="course-col">
            <h3>Intermediate</h3>
            <p>BBA- Bachelor of Business Administration.
                BMS- Bachelor of Management Science.
                BFA- Bachelor of Fine Arts.
                BEM- Bachelor of Event Management.
                Integrated Law Course- BA + LL.B.
                BJMC- Bachelor of Journalism and Mass Communication.
                BFD- Bachelor of Fashion Designing.
                BSW- Bachelor of Social Work.
            </p>
        </div>
        <div class="course-col">
            <h3>Degree</h3>
            <p>Bachelor's in mass communication or Journalism BJMC. ...
                Bachelor's in medicine or dental MBBS/BDS. ...
                B.Sc. ...
                Bachelor of Pharmacy and Master of Pharmacy B. ...
                Bachelor of education or Master's of Education B. ...
                Bachelor of Commerce and Master of Commerce B.Com/M.Com
            </p>
        </div>
        <div class="course-col">
            <h3>Post Graduation</h3>
            <p>Bachelor's in mass communication or Journalism BJMC. ...

```

```

        Bachelor's in medicine or dental MBBS/BDS. ...
        B.Sc. ...
        Bachelor of Pharmacy and Master of Pharmacy B. ...
        Bachelor of education or Master's of Education B. ...
        Bachelor of Commerce and Master of Commerce B.Com/M.Com
    </p>
</div>
</div>
</section>

```

CSS

```

.course{
    width: 80%;
    margin: auto;
    text-align: center;
    padding-top: 100px;
}
h1{
    font-size: 36px;
    font-weight: 600;
}
p{
    color: #777;
    font-size: 14px;
    font-weight: 300;
    line-height: 22px;
    padding: 10px;
}

.row{
    margin-top: 5%;
    display: flex;
    justify-content: space-between;
}
.course-col{
    flex-basis: 31%;
    background: #fff3f3;
    border-radius: 10px;
    margin-bottom: 5%;
    padding: 20px 12px;
    box-sizing: border-box;
}
h3{
    text-align: center;
    font-weight: 600;
    margin: 10px 0;
}

```

```

}
.course-col:hover{
    box-shadow: 0 0 20px 0px rgba(0,0,0,0.2);
}
@media(max-width: 700px){
    .row{
        flex-direction: column;
    }
}

```

• CONTACT COMPONENT

HTML

```

<section class="location">
    <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d248756.1167537794!2d80.06892521258872!3d13.047487786031711!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3a5265ea4f7d3361%3A0x6e61a70b6863d433!2sChennai%2C%20Tamil%20Nadu!5e0!3m2!1sen!2sin!4v1635925203734!5m2!1sen!2sin" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy"></iframe>

</section>

<section class="contact-us">

    <div class="row">
        <div class="contact-col">
            <div>
                <i class="fa fa-home"></i>
                <span>
                    <h5>Visit us</h5>
                    <p>XYZ ROAD, LONDON , UK</p>

                </span>
            </div>
            <div>
                <i class="fa fa-phone"></i>
                <span>
                    <h5>Call Us </h5>
                    <p>
                        +12 8656 98766 <br>
                        Monday to Saturday 10am to 6pm
                    </p>

                </span>
            </div>
            <div>
                <i class="fa fa-envelope-o"></i>
                <span>

```

```

        <h5>E-mail us your query</h5>
        <p>info@eduford.in</p>

    </span>
</div>
</div>
<div class="contact-col">

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

```

CSS

```

.location{
    width: 80%;
    margin: auto;
    padding: 80px 0;
}

.location iframe{
    width: 100%;
}

.contact-us{
    width: 80%;
    margin: auto;
}

.contact-col{
    flex-basis: 48%;
    margin-bottom: 30px;
}

.contact-col div{
    display: flex;
    align-items: center;
    margin-bottom: 40px;
}

.contact-col div.fa{
    font-size: 28px;
    color: #f44336;
    margin: 10px;
    margin-right: 30px;
}

.contact-col div p{
    padding: 0;
}

```

```
.contact-col div h5{
  font-size: 20px;
  margin-top: 5px;
  color: #555;
  font-weight: 400;
}
```

• LOGIN 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" style="text-
align: center;">Log In</div>
      </div>
      <div style="padding-top:30px" class="panel-body" >
        <p style="text-align: center;">
          {{msg}}
        </p>

        <form #f=ngForm autocomplete="off" (submit)="doLogin(f)">

          <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" ngModel #userphone="ngModel" name="uphone" placeholder="Mobile Numb
er" pattern="^(0|[1-9])[0-
9]*)$" required minlength="10" >
          </div><br>

          <p class="alert alert-danger"
*ngIf="userphone.errors?.pattern">
            Please enter valid numbers
          </p>
          <p class="alert alert-danger"
*ngIf="userphone.errors?.minlength">
            Please enter minimum 10 characters
          </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" ngModel #userpassword="ngModel" name="upassword" placeholder="Passwo
rd" required>
    </div>
    <div style="margin-top:10px" class="form-group">
        <!-- Button -->
        <div class="col-sm-12" style="margin-
top: 5px; margin-bottom: 50px; text-align: center;">
            <input type="submit" class="btn btn-
success" value="Login"> &nbsp;
            Don't have Account? <a routerLink="/register
">Sign-up</a>
        </div>
        <div style="text-align: center; margin-
top: 50px; margin-left: 25px;" class="input-group-addfb">
            <a href="https://www.facebook.com/" class=
"btn btn-primary" role="button" target="blank">Login with Facebook</a>
        </div>
    </div>
</form>
<br> <br><br>
</div>
</div>
</div>
</div>

```

TYPE SCRIPT

```

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

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

  msg!: string;

  constructor(public userSer : UsersService, public myRouter : Router) { }

```

```

ngOnInit(): void {
}

doLogin(form: NgForm)
{
  console.log(form.value);

  this.userService.doUserLogin(form.value).subscribe((data: any[]) => {

    console.log(data);

    // this.msg = data;

    if (data.length == 0)
    {
      this.msg = "Invalid Login";
    }
    else {
      localStorage.setItem("loggeduser", data[0]._id);
      this.myRouter.navigateByUrl("/court");
    }

  }, (error: any) => {

    console.log(error);

    this.msg = "Something went wrong";

  }));
}
}

```

• REGISTER 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" style="text-
align: center;">Register</div>
      </div>
      <span>
        {{msg}}
      </span>
      <div style="padding-top: 30px" class="panel-body" >

```

```

        <form (submit)=userRegistration(f) #f="ngForm" autocomplete="off">
            <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 #username="ngModel" placeholder="Username" required minlength="5">
            </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">
                Please enter minimum 5 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 your E-mail
            </p>
            <p class="alert alert-danger" *ngIf="useremail.errors?.email">
                Please enter valid E-mail
            </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 #userpwd="ngModel" placeholder="Password" minlength="6" required>
            </div>
            <p class="alert alert-danger" *ngIf="userpwd.errors?.required && userpwd.touched">
                Please enter your password
            </p>
            <p class="alert alert-danger"

```

```

*ngIf="userpwd.errors?.minlength">
    Password must contain 6 characters
</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="urepassword" ngModel #repwd="ngModel" placeholder="Re enter pa
ssword" required>
</div>
<p class="alert alert-danger"
*ngIf="repwd.errors?.required && repwd.touched
">
    Password must match pre-entered password
</p>

    <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 #userphone="ngModel" placeholder="Phone Number
" pattern="^(0|[1-9][0-
9]*)$" minlength="10" required >
    </div>
    <p class="alert alert-danger"
*ngIf="userphone.errors?.required && userphone
.touched">
        Please enter phone number
    </p>
    <p class="alert alert-danger"
*ngIf="userphone.errors?.pattern">
        Please enter valid numbers
    </p>
    <p class="alert alert-danger"
*ngIf="userphone.errors?.minlength">
        Enter 10 numbers
    </p>

    <div style="margin-top:10px" class="form-group">
        <!-- Button -->
        <div class="col-sm-12">
            <input type="submit" class="btn btn-
success" value="Register"/> &nbsp;

```

```

                Already have Account? <a routerLink="/login"
>Login</a>
            </div>
        </div>
    </form>
    <br> <br><br>

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

```

TYPE SCRIPT

```

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 {
  }
  userRegistration(form:NgForm)
  {
    console.log("User Registered");
    console.log(form.value);

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

      console.log(data);

      this.msg = data;

      form.reset();

    }, (error:any )=>{

      console.log(error);
    }
  )
}

```

```

        this.msg = "Something Went Wrong!!";

    });

}

}

```

• STUDENT DETAILS COMPONENT

HTML

```

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

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

    <div class="col-sm-4">
      Search: <input type="text" #search (keyup)="doSearch(search.value)"
"> &nbsp;

      <a routerLink="/adduser">
        <button class="btn btn-success">Add User</button>
      </a>

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

<div class="col">
  <p>
    {{msg}}
  </p>
  <table class="table table-striped">
    <thead>
      <tr>
        <th scope="col" style="color: crimson; font-size: x-large;">#</th>
        <th scope="col" style="color: crimson; font-size: x-
large;" >Name</th>
        <th scope="col" style="color: crimson; font-size: x-
large;">Email</th>
        <th scope="col" style="color: crimson; font-size: x-
large;">Phone</th>
        <th scope="col" style="color: crimson; font-size: x-
large;">Operations</th>
      </tr>

```

```

        </thead>
        <tbody>
            <tr *ngFor="let user of users; index as i;">
                <th scope="row">{{i+1}}</th>
                <td>{{user.uname}}</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>
    <p *ngIf="users.length==0">
        No Records Found
    </p>
</div>

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

```

TYPE SCRIPT

```

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 {

    msg!: string;

    users : any[] = [];

    constructor(public userSer : UsersService) { }

    ngOnInit(): void {

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

```

```

        console.log(data);

        this.users = data;

    }, (error:any)=>{

        this.msg = "Something went wrong";
    });
}

deleteUser(userid:number)
{
    if(confirm("Are you sure want to delete this users?"))
    {
        // console.log("User deleted with 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)=>{

            this.msg = "Something went ";
        })
    }
}

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

        console.log(data);

        this.users = data;

    }, (error:any)=>{

        console.log(error);

    });
}

```

```
}
}
}
```

• ADD USER 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">Add Users</div>
      </div>
      <div style="padding-top:30px" class="panel-body" >
        <p>
          {{msg}}
        </p>
        <form #f="ngForm" (submit)="addUser(f)">

          <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 #username="ngModel" (blur)="usernameCheck(usern
ame.value)" placeholder="Username" required minlength="6">

            </div>

            <p class="alert alert-danger"
*ngIf="username.errors?.required && username.t
ouched">

              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 #useremail="ngModel" placeholder="Email" requi
red email>

              </div>

              <p class="alert alert-danger"
*ngIf="useremail.errors?.required && useremail.tou
ched">

                Please Enter Your Email
```

```

        </p>
        <p class="alert alert-danger"
        *ngIf="useremail.errors?.email && useremail.touche
d">
            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 #userpassword="ngModel" placeholder="Passw
ord" 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 placeholder="Phone Number" #userphone="ngMode
l" 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="Add User" [disabled]="f.invalid || !usernameAvail" /> &nbsp;
            </div>
        </div>
    </form>
    <br> <br><br>
</div>
</div>
</div>

```

TYPE SCRIPT

```

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 {
  }

  addUser(form:NgForm)
  {
    this.userSer.doUserRegistration(form.value).subscribe((data:string)=>{

      console.log(data);

      this.msg = data;

      form.reset();

    }, (error:any)=>{

      console.log(error);

      this.msg = "Something Went Wrong!!";

    });
  }

  usernameCheck(username:string)
  {
    this.userSer.usernameCheckAvailability(username).subscribe((data:any[])=>{

      console.log(data);

      if(data.length==0)
      {
        this.msg = "Congrats Username Available for You!!";

        this.usernameAvail = true;
      }
      else {

```

```

        this.msg = "Username Already Taken!!";
        this.usernameAvail = false;
    }

    }, (error:any)=>{

        console.log(error);

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

```

• EDIT USER 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" placeho
lder="Username" required minlength="6">

                    </div>

                    <p class="alert alert-danger"
*ngIf="username.errors?.required && username.t
ouched">

                        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" place
holder="Email" required email>
        </div>

        <p class="alert alert-danger"
*ngIf="useremail.errors?.required && useremail.tou
ched">

            Please Enter Your Email
        </p>
        <p class="alert alert-danger"
*ngIf="useremail.errors?.email && useremail.touche
d">

            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="ngMod
el" 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>

</div>
```

TYPE SCRIPT

```
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 {

  msg!: string;

  userdata!: { _id: number; uname: string; uemail: string; upassword: string;
  uphone: string; };

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

  ngOnInit(): void {

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

      console.log(param.userid);

      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;
    console.log(form.value);

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

        this.msg = data;

    }, (error:any)=>{

        this.msg = "Something went wrong!!";
    });
}
}

```

• ANGULAR GUARD COMPONENT

```

import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { UsersService } from './users.service';

@Injectable({
    providedIn: 'root'
})
export class AuthGuard implements CanActivate {

    constructor(public userSer : UsersService, public myRouter : Router)
    {

    }

    canActivate(): boolean {

        if(!this.userSer.isLoggedIn())
        {
            this.myRouter.navigateByUrl("/");
        }

        return this.userSer.isLoggedIn();
    }
}

```

```
}
```

• APP ROUTING COMPONENT

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AboutComponent } from './about/about.component';
import { AdduserComponent } from './adduser/adduser.component';
import { ContactComponent } from './contact/contact.component';
import { CourseComponent } from './course/course.component';
import { EdituserComponent } from './edituser/edituser.component';
import { HomeComponent } from './home/home.component';
import { LoginComponent } from './login/login.component';
import { RegisterComponent } from './register/register.component';
import { UsersComponent } from './users/users.component';

const routes: Routes = [
  {path:"register",component: RegisterComponent},
  {path: "login",component: LoginComponent},
  {path: "users",component: UsersComponent},
  {path: "adduser", component: AdduserComponent},
  {path: "edituser/:userid", component: EdituserComponent},
  {path:"", component:HomeComponent},
  {path:"about", component:AboutComponent},
  {path:"course", component:CourseComponent},
  {path:"contact", component:ContactComponent},

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

• PORT 3000 CONNECTION

```
const http = require("http");
const app = require("./backend/app");

//const server = http.createServer((req,res)=>{

  // res.sendDate("Welcome to my list Node Js");

//});
```

```
server.listen(3000,()=>{
    console.log("Check port no 3000");
});
```

```

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

C:\Users\Esther Maglin\angular>ng serve -o
  Browser application bundle generation complete.

Initial Chunk Files | Names | Size
vendor.js           | vendor | 2.88 MB
styles.css, styles.js | styles | 524.29 kB
polyfills.js        | polyfills | 510.57 kB
main.js             | main | 120.69 kB
runtime.js          | runtime | 6.61 kB

| Initial Total | 3.93 MB

Build at: 2021-11-12T03:59:50.949Z - Hash: 57ea762335a0cbc2caa0 - Time: 786/ms

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

  Compiled successfully.
  Browser application bundle generation complete.

Initial Chunk Files | Names | Size
main.js            | main | 119.35 kB

4 unchanged chunks

Build at: 2021-11-12T04:16:32.306Z - Hash: 6c62cfe0c1622aa352 - Time: 1383ms

  Compiled successfully.
  Browser application bundle generation complete.

Initial Chunk Files | Names | Size
main.js            | main | 118.35 kB

4 unchanged chunks

Build at: 2021-11-12T04:17:31.216Z - Hash: d014633e072be96837a9 - Time: 1233ms

  Compiled successfully.

```

```
C:\Windows\System32\cmd.exe - nodeemon server.js
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Esther Maglin\angular>nodeemon server.js
[nodeemon] 2.0.14
[nodeemon] to restart at any time, enter 'rs'
[nodeemon] watching path(s): *.*
[nodeemon] watching extensions: js,mjs,json
[nodeemon] starting 'node server.js'
check port no 3000
(node:9219) [DEP0008 DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { unifiedTopology: true } to the MongoClient constructor.
(Use 'node --trace-warnings ...' to show where the warning was created)
DB Connected
Middleware 1
{
  username: 'Esther',
  uemail: 'esther@gmail.com',
  upassword: 'esther',
  urepassword: 'esther',
  uphone: '6383148787',
  _id: 163668979355
}
Middleware 1
{ uphone: '6383148787', upassword: 'esther' }
Middleware 1
{ user: 'Esther', uphone: '6383148787', upassword: 'esther' }
Middleware 1
{ userid: '1636457633603' }
Middleware 1
{
  username: 'Esther',
  uemail: 'esther@gmail.com',
  upassword: 'esther',
  uphone: '6383148787',
  _id: 1636457633603
}
```