

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
Sri Swetha. G
(Reg. No. 39120120)



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
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 **SRI SWETHA .G** (Reg. No: 39120120) who carried out the project entitled “**Badminton Court Booking App Using Angular JS**” under my supervision from September 2021 to November 2021.

Internal Guide

Dr. J. JABEZ, 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, SRI SWETHA.G (Reg. No. 39120120) hereby declare that the Project report entitled **"Badminton Court Booking App Using Angular JS"** done by me under the guidance of **Dr. J.JABEZ, M.E., Ph.D**, is submitted in partial fulfillment of the requirements for the award of Bachelor of 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 INSTITUTE OF SCIENCE AND TECHNOLOGY** 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. SUBHASHINI, M.E., Ph.D., Head of the Department, Dept. 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. J.JABEZ, 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 Badminton Court Booking App, the player who has account already can book the court according to their needs, and those who are not having the account can register themselves and can be able to login. Once after they login, they have to select the court timings and they will be given with some membership plans, and they can select the membership according to their afford. The next step is the players must agree with all terms and conditions of the management and thereby their booking will be done. Once after they booked the court, the player can logout, and they will be directed to the login page again. Another feature of this app is admin user 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 players will be displayed in the mongo database. The admin actions will not only be carried out in front end, but also in backend i.e. in mongo database.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	i
	LIST OF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1.	INTRODUCTION	1
	WHAT IS ANGULAR JS?	1
	ROLE OF HTML, CSS & JS IN ANGULAR	2
	REQUIREMENTS FOR ANGULAR PROJECT	3
2.	BADMINTON COURT BOOKING APP	4
	OVERVIEW	4
	FUNCTIONAL REQUIREMENTS	4
	COMPONENTS	5
	OVERVIEW OF COMPONENTS	5
	BACKEND	16
3.	SUMMARY AND CONCLUSIONS	17
	SUMMARY	17
	CONCLUSIONS	17
	REFERENCES	18
	APPENDIX	18
	A. SOURCE CODE	18
	B. SCREENSHOTS	42

LIST OF FIGURES

FIGURE No.	TITLE	PAGE No.
1.1	Login Page	6
1.2	Register Page	7
1.3	Court Registration Page	8
1.4	Membership Page	9
1.5	Terms and Conditions Page	10
1.6	User Management Page	11
1.6.1	Add Player Page	12
1.6.2	Edit Player Page	13
1.6.3	Search Player Page	13
1.6.4	Delete Player Page	14
1.7	Logout Page	15
1.8	Mongodb connection	16

LIST OF TABLES

TABLE No.	TITLE	PAGE No.
2.1	Login Fields	6
2.2	Register/Sign-up Fields	7
2.3	Court Registration Fields	8
2.4	Membership Fields	9
2.5	Add Player Fields	11
2.6	Edit Player Fields	12

LIST OF ABBREVIATIONS

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

INTRODUCTION

What 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 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.

AngularJS's 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.

AngularJS 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. 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.

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.

BADMINTON COURT BOOKING APP

Overview:

This project aims to maintain the information about the players those who are in need of booking badminton court. It will help the court authorities to manage details of the player. Court Admin will be able to register themselves and search data of the players using the search bar and for the admin purpose, user management portal will be also available.

Only admin will be able to update players' information. There are many components like login, membership etc. Once player register themselves, they will be directed to various fields and finally the booking will be done. This app is done with HTML, CSS, JS and the main part is the angular js.

The Badminton Court Booking App designed by Team Phoenix will be very easy and a delicate for the player to book their time slot. Now a days, internet is the most widely used network in modern world and no one is preferring to go out and booking their slots. For their easy purpose, Team Phoenix made this Badminton Court Booking App, so that users can book by sitting in their home itself.

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 / Sign-up
- ❖ Court Registration
- ❖ Membership
- ❖ Terms and Conditions
- ❖ User Management Actions (for admin purpose)
 - Add User
 - Edit User
 - Search User
 - Delete User
- ❖ Logout

OVERVIEW OF COMPONENTS:

- **Login portal:**

The login portal is used for the registered players to login into the app. The admin also can login to do the user management related activities.

FIELDS	VALIDATIONS
Phone No	Required, Minlength=10
Password	Required

Table 2.1: Login Fields

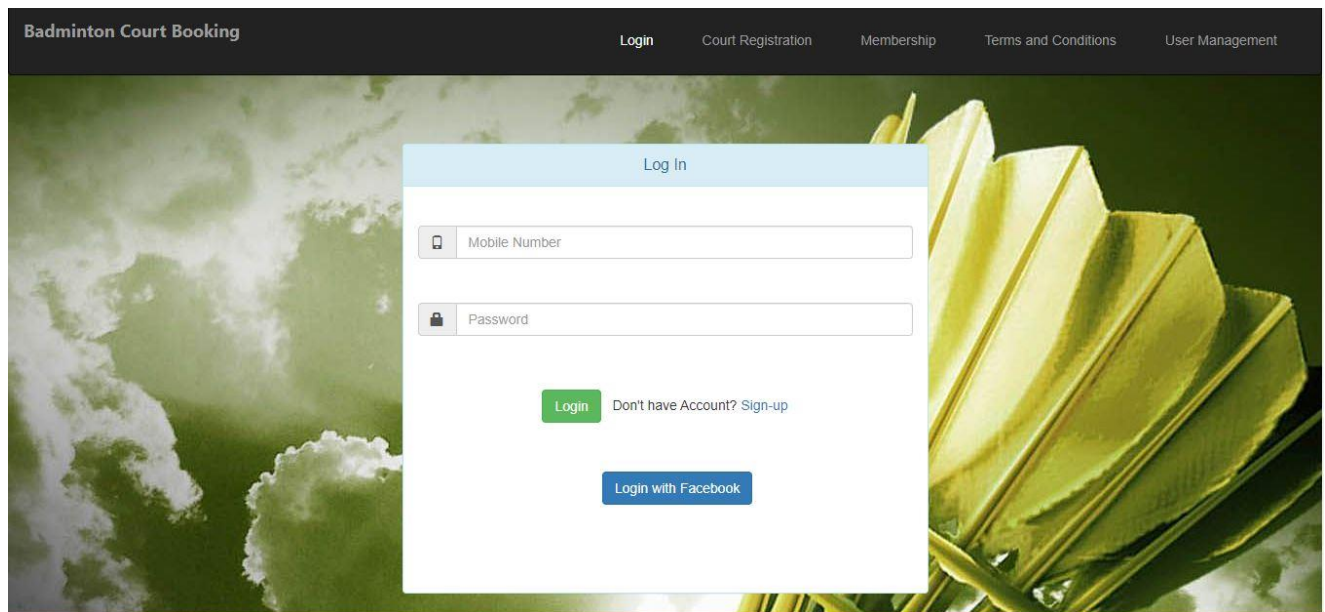


Fig 1.1: Login Page

- **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	VALIDATIONS
Username	Required, Minlength = 5
E-mail	Required, E-mail validation
Password	Minlength = 6
Retype password	Same as password
Phone No	Number validation, Minlength = 6

Table 2.2: Register / Sign-up fields

Badminton Court Booking

Login Court Registration Membership Terms and Conditions User Management

Register

Username

Email

Password

Re enter password

Phone Number

Register Already have Account? Login

Copyright 2021 | All Rights Reserved

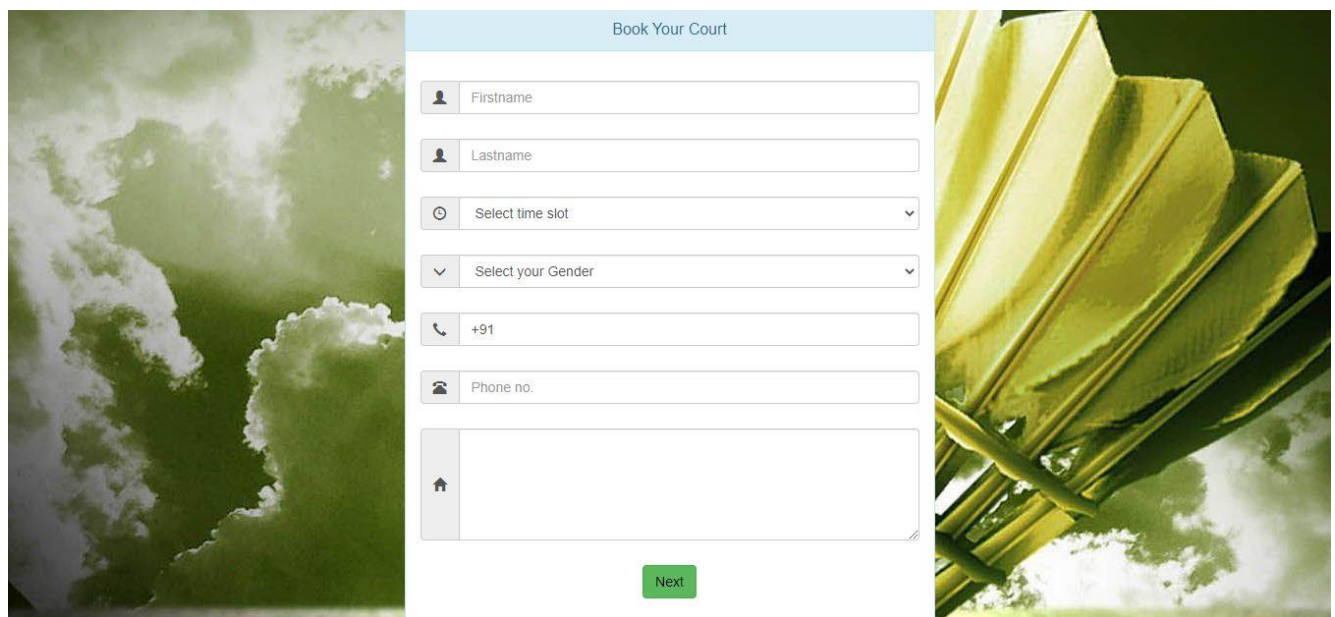
Fig 1.2: Register Page

- **Court Registration portal:**

In court registration portal, the players can book their playing time slot acc to their convenience. By giving all information about them, they can confirm their time slot and they must enter their detail.

FIELDS	VALIDATIONS
First Name	Required
Last Name	Required
Time Slot	Required
Gender	Required
Country Code	+91 default
Phone No	Required
Current Address	Required

Table 2.3: Court Registration Fields



The screenshot shows a web form titled "Book Your Court". The form is set against a background of a cloudy sky. The form fields are as follows:

- Firstname:** A text input field with a person icon on the left.
- Lastname:** A text input field with a person icon on the left.
- Select time slot:** A dropdown menu with a clock icon on the left.
- Select your Gender:** A dropdown menu with a downward arrow icon on the left.
- +91:** A text input field with a telephone icon on the left.
- Phone no.:** A text input field with a telephone icon on the left.
- Current Address:** A large text area with a house icon on the left.

A green "Next" button is located at the bottom center of the form.

Fig 1.3: Court Registration Page

- **Membership Portal:**

This portal allows player to select their membership for the payment purpose of court booking. Here the plans like silver, standard and gold are given so that the player can pay the amount acc to the plan they've selected.

FIELDS	VALIDATIONS
Name	Required
Select Membership Type	Required
Phone No	Required
Checkbox for agree	Required

Table 2.4: Membership Fields

Fig 1.4: Membership Page

- **Terms and Conditions:**

In this portal, the player must agree to the terms and conditions thrown by the admin and then only the players' booking gets officially confirmed.

Badminton Court Booking

Court Registration Membership **Terms and Conditions** User Management Logout

Our Terms and Conditions

1. Everyone who uses the Phoenix premises for any sports related activity (racket play or fitness activity) is required to have an active membership or register as a guest.
2. Phoenix reserves the right to limit the number of Guests and badminton court rental players who are permitted use the facilities.
3. Phoenix shall not be held responsible for vehicles or any damage to said vehicles while using the parking areas within the premises.
4. Phoenix staff may from time to time, inform you about additional rules and regulations or changes, if any.
5. Phoenix has instituted a cell phone policy for the protection of each Member's privacy and safety as well as to maintain a peaceful environment for all to enjoy. Cell phone use is prohibited in all facilities. Common social areas such as parking space and open spaces may be used for conversations. We appreciate your adherence to this policy on behalf of your fellow members.
6. Children under 12 years of age must have parental supervision at all times, on and off the courts. Children must not, at any time, enter or play in the parking area. If so, Phoenix will not be liable for any incidents.
7. Absolutely no food or drinks are allowed on the floor court areas except water. Anyone who violates this policy will have to dispose the food or drinks immediately or be required to leave Phoenix premises immediately.
8. All members and guests who use any of the Phoenix courts realize and accept that badminton is a vigorous sport and is possible to sustain injuries when playing. All members and guests play badminton at their own risk. The academy shall not provide any emergency medical assistance or transportation to medical facility.
9. Non players, including children are not permitted to be on the courts at any time.
10. Phoenix does not allow smoking, spitting, gum chewing, animals or weapons in the facility.
11. No photography or taking of videos are permitted.
12. Any one found to be violating this policy will be required to leave the Phoenix premises immediately without refund and may be barred from entering the Phoenix premises in the future.

DRESS CODE :

1. Only Non-marketing badminton shoes are permitted inside the court. No barefoot play is permitted.

DRESS CODE :

1. Only Non-marketing badminton shoes are permitted inside the court. No barefoot play is permitted.
2. All players must wear a T-shirt at all times while on the courts. Shorts of jean material are not permitted.
3. Black soled shoes of any kind are not allowed.
4. Shoes to be worn on the courts are to be brought into the club and players will change into those shoes prior to entering the court area. Regardless of the footwear type, vest, dirty or shoes that have been worn outside will not be permitted on the courts. Likewise a player who leaves the court to go to the parking lot will not be allowed to wear those shoes back on to the courts.
5. Street shoes, slippers, flip flops or any kind of footwear not specifically designed for badminton is not permitted.
6. No Street clothing. Sleeveless T- Shirts, tops with vulgar language or defamatory logos or designs.
7. Everyone needs to bring a towel and full water bottle.

PERSONAL CONDUCT :

1. Phoenix members, guests and court rental guests (herein referred to as 'players') are subject to the rules and guidance of the Phoenix staff

PERSONAL CONDUCT :

1. Phoenix members, guests and court rental guests (herein referred to as 'players') are subject to the rules and guidance of the Phoenix staff and must wholly respect and follow staffs instructions and club regulations at all times while playing in the club.
2. Players agree to conduct themselves in a quiet and well-mannered fashion when in the club and not cause any disturbances or interfere with the safe use or enjoyment of the academy by other players.
3. Members and Guests shall restrain themselves from using profanities while in the premises.
4. Members and Guests shall refrain from physical contact with another person in any angry or threatening way. Harassment or intimidation by words, gestures, body language or any other menacing behaviour.

☐ I hereby agree to abide by the terms and conditions as provided by the Phoenix Board of Badminton Council.

[Book Court](#)

Copyright 2021 | All Rights Reserved

Fig 1.5: Terms and Conditions Page

- **User Management Portal:**

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.

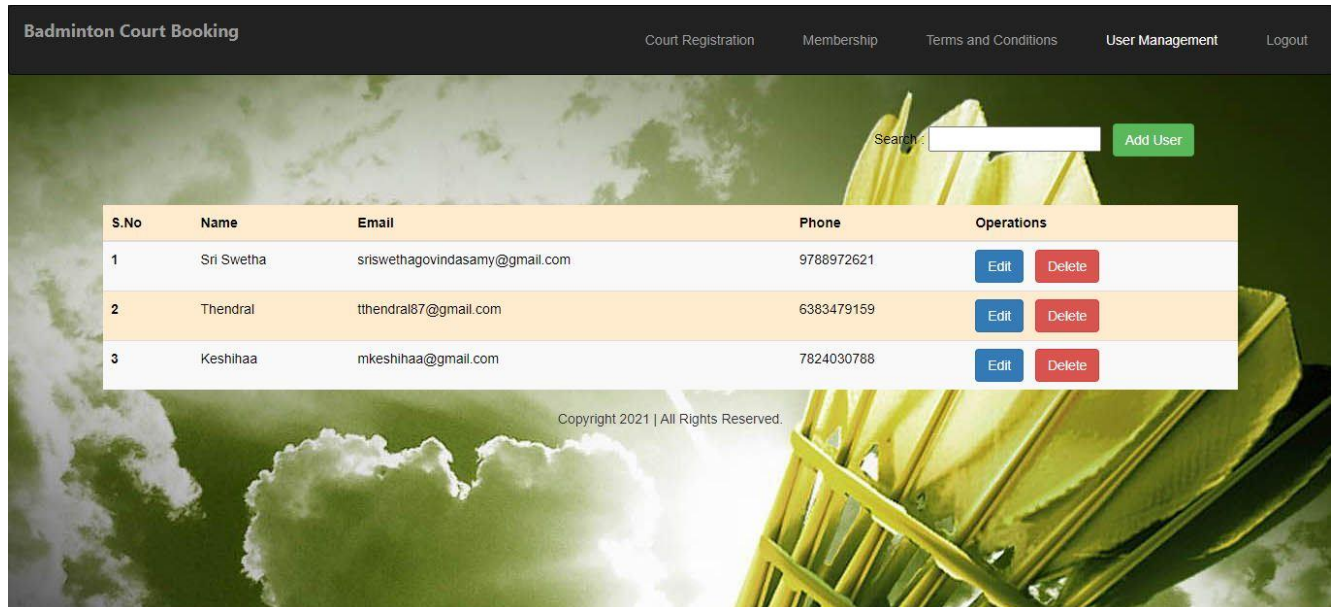


Fig 1.6: User Management Page

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

- **Add User Portal:**

Add user component mainly involves the admin action of adding player apart from user registration.

FIELDS	VALIDATIONS
Username	Required, Minlength = 6
E-mail	Required, E-mail validation
Password	Required, Minlength = 6
Phone No	Pattern validation, Minlength = 6

Table 2.5: Add Player Fields

Fig 1.6.1: Add Player Page

▪ Edit User Portal:

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

FIELDS	VALIDATIONS
Username	Username gets auto-updated
E-mail	E-mail gets auto-updated
Password	Password gets auto-updated
Phone No	Phone No gets auto-updated

Table 2.6: Edit Player Fields

Badminton Court Booking

Court Registration Membership Terms and Conditions User Management Logout

Edit User

Update

Copyright 2021 | All Rights Reserved

Fig 1.6.2: Edit Player Page

▪ Search User Option :

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

Badminton Court Booking

Court Registration Membership Terms and Conditions User Management Logout

Search:

S.No	Name	Email	Phone	Operations
1	Sri Swetha	sriswethagovindasamy@gmail.com	9788972621	Edit Delete

Copyright 2021 | All Rights Reserved

Badminton Court Booking

Court Registration Membership Terms and Conditions User Management Logout

Search:

S.No	Name	Email	Phone	Operations
No Records Found				

Copyright 2021 | All Rights Reserved

Fig 1.6.3: Search Player Page

- **Delete User Option:**

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

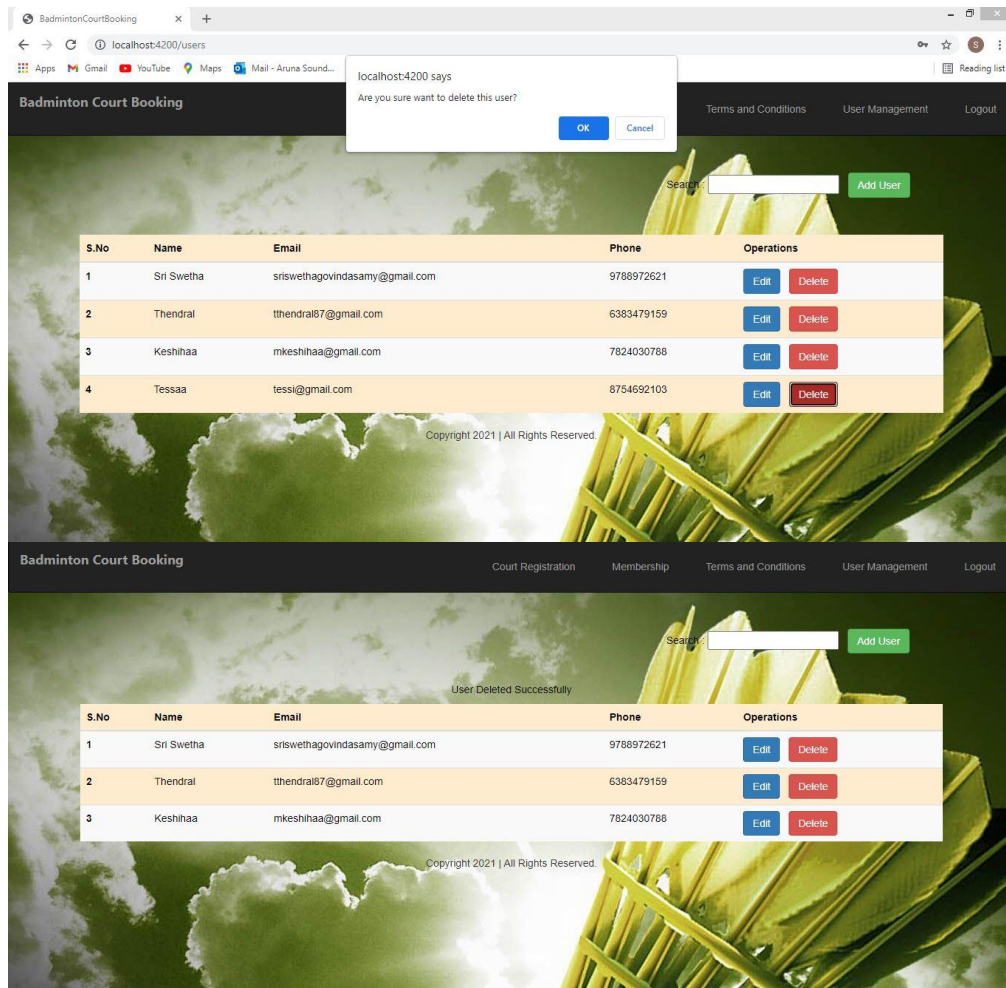


Fig 1.6.4: Delete Player Page

- **Logout:**

After completing all parts of booking, the booking will be confirmed. Once after the booking gets confirmed, the user can logout from 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 book court. And this is the final stage of court booking process!!

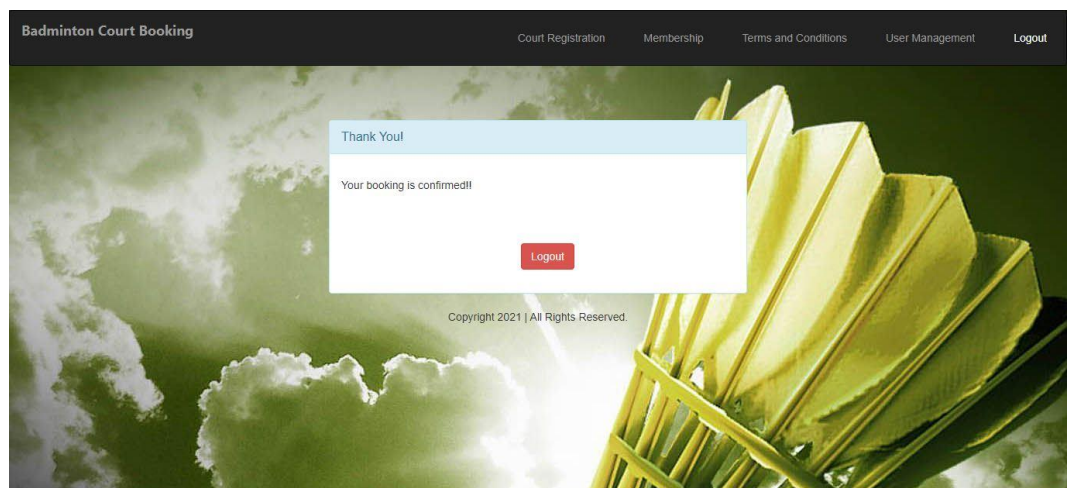


Fig 1.7: Logout Page

- **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 responses of the server.

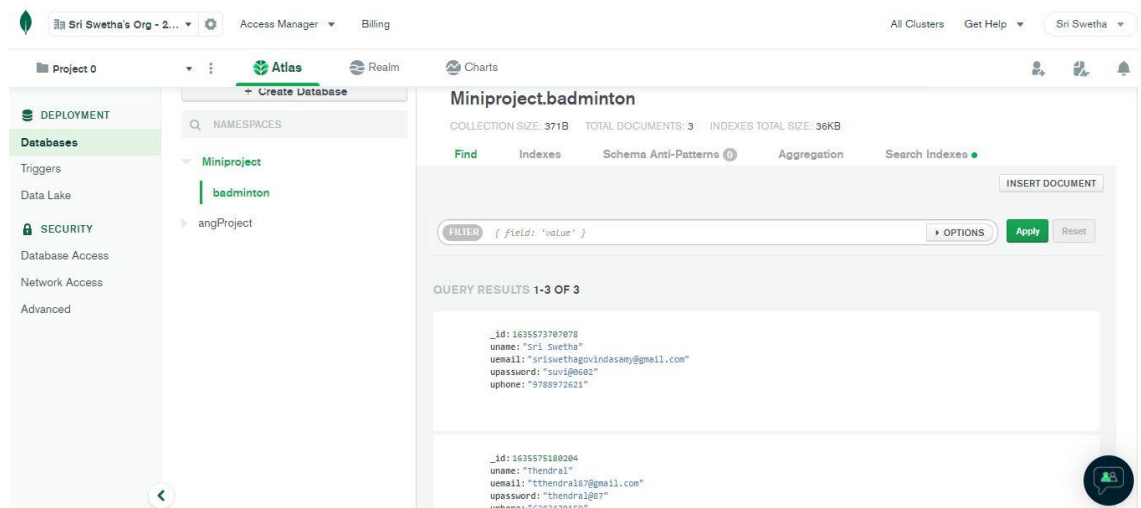


Fig 1.8: Mongodb Connection

SUMMARY AND CONCLUSIONS

- **SUMMARY:**

This badminton court booking app helps many people those who're more interested in playing badminton. Now a days in the modern lifestyle, problems related to health issues arises which makes the people hospitalize. So in order to maintain a healthy lifestyle, regular exercises and sports related activity is important. Therefore I created "Badminton Court Booking App" in which the booking will be done in fast manner and in an affordable membership plans. 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 booking!!!

- **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.

REFERENCES:

PLAYCO – An app that can be downloaded from google play store has been taken as 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());
mongodb.connect("mongodb+srv://swetha:govindasamy@cluster0.hlo7f.mongodb.net/Miniproject?retryWrites=true&w=majority", (error, result)=>{

    if(error)
    {
        console.log("DB not connected");
    }
    else{
        db = result.db("Miniproject")
        console.log("DB connected successfully");
    }
});

app.use((req, res, next)=>{

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

});

app.use("/home", (req, res, next)=>{

    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("Welcome to Phoenix Badminton Council");
    });
    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("badminton").insertOne(req.body, (error,data)=>{
            if(error)
            {
                res.json(403).json("Error In Inserting Document");
            }
            else
            {
                res.json("User Registered Successfully");
            }
        })
    });

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

        console.log(req.body);
        db.collection("badminton").find(req.body, {projection: {_id:1,uname:1}}).toArray((error, data)=>{

            if(error)
            {
                res.json(403).json("Error In Finding Document");
            }
            else
            {
                res.json(data);
            }

        });

    });

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

        db.collection("badminton").find(null, {projection: {upassword : 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("badminton").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("badminton").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("badminton").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("badminton").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 serachCond = {uname : search};
    }
    else
    {
        var serachCond = null;
    }

    db.collection("badminton").find(serachCond).toArray((error, data)=>{
        res.json(data);
    });
});

module.exports = app;

```

- **HEADER COMPONENT:**

HTML:

```

<style>
    .navbar-brand{
        font-family:-apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Cantarell, 'Open Sans',
        'Helvetica Neue', sans-serif;
    }
</style>

<nav class="navbar navbar-inverse">
    <div class="container-fluid">
        <div class="navbar-header">
            <a class="navbar-brand"><b>Badminton Court Booking</b></a>
        </div>
        <ul class="nav navbar-nav navbar-right">
            <li style="color: white; padding: 10px;"><a routerLink="/"
                *ngIf="!userSer.isLoggedIn()">Login</a></li>
            <li style="color: white; padding: 10px;"><a routerLink="/court">Court Registration</a></li>
            <li style="color: white; padding: 10px;"><a routerLink="/member">Membership</a></li>
            <li style="color: white; padding: 10px;"><a routerLink="/terms">Terms and Conditions</a></li>
            <li style="color: white; padding: 10px;"><a routerLink="/users">User Management</a></li>
            <li style="color: white; padding: 10px;"><a routerLink="/booked"
                *ngIf="userSer.isLoggedIn()">Logout</a></li>
        </ul>
    </div>
</nav>

```

- 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 Number" 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="Password" 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>
</div>
</div>

```

TYPESCRIPT:

```

import { Component, OnInit } from '@angular/core';
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(frmVal: any)
  {
    console.log(frmVal.value);
    this.userSer.doUserLogin(frmVal.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");
      }
      frmVal.reset();

    }, (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>
      <div style="padding-top:30px" class="panel-body" >
        <span>
          {{msg}}
        </span>

        <form (submit)=doRegister(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 password" 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</a>
            </div>
        </div>
    </form>
    <br> <br><br>

    </div>
</div>

```

TYPESCRIPT:

```

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

@Component({
    selector: 'app-register',

```

```

    templateUrl: './register.component.html',
    styleUrls: ['./register.component.css']
  })
  export class RegisterComponent implements OnInit {

    msg : string | undefined ;

    constructor(public userSer : UsersService) { }

    ngOnInit(): void {
    }
    doRegister(frmVal:any)
    {
      console.log("User Registered");
      console.log(frmVal.value);
      this.userSer.doUserRegistration(frmVal.value).subscribe((data:string)=>{
        console.log(data);
        this.msg = data;
        frmVal.reset();
      }, (error:any)=>{

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

      });

    }
  }
}

```

- **COURT REGISTRATION 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;">Book Your Court</div>
      </div>
      <div style="padding-top:30px" class="panel-body" >
        <form>
          <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="firstname" placeholder= "Firstname"
required>

          </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="lastname" placeholder="Lastname"
size="15" required >

          </div>

```

```

<div style="margin-bottom: 25px" class="input-group">
  <span class="input-group-addon"><i class="glyphicon glyphicon-time"></i></span>
    <select class="form-control">
      <option value="#">Select time slot</option>
      <option value="5.30-7.00">5.30-7.00</option>
      <option value="7.30-9.00">7.30-9.00</option>
      <option value="9.00-11.30">9.00-11.30</option>
      <option value="12.00-2.00">12.00-2.00</option>
      <option value="2.30-4.00">2.30-4.00</option>
      <option value="4.30-6.00">4.30-6.00</option>
      <option value="8.30-10.30">8.30-10.30</option>
    </select>

  </div>

<div style="margin-bottom: 25px" class="input-group">
  <span class="input-group-addon"><i class="glyphicon glyphicon-menu-down"></i></span>
    <select class="form-control">
      <option value="#">Select your Gender</option>
      <option value="Male">Male</option>
      <option value="Female">Female</option>
      <option value="Others">Others</option>
    </select>
  </div>

<div style="margin-bottom: 25px" class="input-group">
  <span class="input-group-addon"><i class="glyphicon glyphicon-
earphone"></i></span>
    <input type="text" name="country code" class="form-control"
placeholder="Country code" value="+91" size="2" required>
  </div>
  <div style="margin-bottom: 25px" class="input-group">
    <span class="input-group-addon"><i class="glyphicon glyphicon-phone-alt"></i></span>
    <input type="text" name="phone" class="form-control" placeholder="Phone no."
size="10" required>
  </div>
  <div style="margin-bottom: 25px" class="input-group">
    <span class="input-group-addon"><i class="glyphicon glyphicon-home"></i></span>
    <textarea cols="65" rows="5" class="form-control" placeholder="Current address"
name="address" required>
    </textarea>
  </div>
  <div style="margin-bottom: 25px; text-align: center;" class="input-register">
    <a routerLink="/member" class="btn btn-success" style="color:black"
role="button">Next</a>
  </div>

</form>
<br> <br><br>

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

```

- **MEMBERSHIP 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;">Choose Your Membership</div>
      </div>
      <div style="padding-top:30px" class="panel-body" >
        <form 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="firstname" placeholder= "Name"
required>

          </div>

          <div style="margin-bottom: 25px" class="input-group">
            <span class="input-group-addon"><i class="glyphicon glyphicon-menu-down"></i></span>
            <select class="form-control">
              <option value="#">Select Membership Type</option>
              <option value="standard">Standard - Rs.500/week</option>
              <option value="silver">Silver - Rs.2500/month</option>
              <option value="gold">Gold - Rs.25000/year</option>
            </select>

          </div>

          <div style="margin-bottom: 25px" class="input-group">
            <span class="input-group-addon"><i class="glyphicon glyphicon-phone-alt"></i></span>
            <input type="text" name="phone" class="form-control" placeholder="Phone no."
size="10" required>

          </div>
          <div style="margin-bottom: 25px" class="input-group">
            <input type="checkbox"> I have accepted the membership plan with my
knowledge<br><br>

            <input type="checkbox"> I will not withdraw from my plan in any circumstances
          </div>
          <div style="margin-bottom: 25px; text-align: center;" class="input-register">

            <a routerLink="/terms" class= "btn btn-success" style="color: black"
role="button">Next</a>

          </div>

        </form>
        <br> <br><br>

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

- **TERMS AND CONDITIONS:**

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;">Our Terms and Conditions</div>
      </div>
      <form>
        <div style="margin-bottom: 25px ; margin-right: 20px; margin-top: 20px;" class="input-
group">

          <ol>
            <li>Everyone who uses the Phoenix premises for any sports related activity (racket
play or fitness activity) is required to have an active membership or register as a guest.</li><br>
            <li>Phoenix reserves the right to limit the number of Guests and badminton court
rental players who are permitted use the facilities.</li><br>
            <li>Phoenix shall not be held responsible for vehicles or any damage to said
vehicles while using the parking areas within the permises.</li><br>
            <li>Phoenix staff may from time to time, inform you about additional rules and
regulations or changes, if any.</li><br>
            <li>Phoenix has instituted a cell phone policy for the protection of each Member's
privacy and safety as well as to maintain a peaceful environment for all to enjoy. Cell phone use is prohibited
in all facilities. Common social areas such as parking space and open spaces may be used for conversations. We
appreciate your adherence to this policy on behalf of your fellow members.</li><br>
            <li>Children under 12 years of age must have parental supervision at all times, on
and off the courts. Children must not, at any time, loiter or play in the parking area. If so, Phoenix will not
be liable for any incidents.</li><br>
            <li>Absolutely no food or drinks are allowed on the floor court areas except water.
Anyone who violates this policy will have to dispose the food or drinks immediately or be required to leave
Phoenix premises immediately.</li><br>
            <li>All members and guests who use any of the Phoenix courts realize and accept that
badminton is a vigorous sport and is possible to sustain injuries when playing. All members and guests play
badminton at their own risk, The academy shall not provide any emergency medical assistance or transportation to
medical facility.</li><br>
            <li>Non players, including children are not permitted to be on the courts at any
time.</li><br>
            <li>Phoenix does not allow smoking, spitting, gum chewing, animals or weapons in the
facility.</li><br>
            <li> No photography or taking of videos are permitted.</li><br>
            <li>Any one found to be violating this policy will be required to leave the Phoenix
premises immediately without refund and may be barred from entering the Phoenix premises in the future.</li><br>
          </ol>
          <div style="margin-bottom: 25px ; margin-right: 20px; margin-top: 10px; margin-left:
20px;" class="input-group">
            <h5><b>DRESS CODE :</b></h5><br>
            <ol>
              <li>Only Non-marketing badminton shoes are permitted inside the court. No barefoot
play is permitted. </li><br>
              <li>All players must wear a T-shirt at all times while on the courts. Shorts of jean
material are not permitted.</li><br>
              <li>Black soled shoes of any kind are not allowed.</li><br>
              <li>Shoes to be worn on the courts are to be brought into the club and players will
change into those shoes prior to entering the court area. Regardless of the footwear type, wet, dirty or shoes
that have been worn outside will not be permitted on the courts. Likewise a player who leaves the court to go to
the parking lot will not be allowed to wear those shoes back on to the courts.</li><br>
              <li>Street shoes, slippers, flip flops or any kind of footwear not specifically
designed for badminton is not permitted.</li><br>
            </ol>
          </div>
        </div>
      </form>
    </div>
  </div>
</div>
```

```

        <li>No Street clothing, Sleeveless T- Shirts, tops with vulgar language or
defamatory logos or designs.</li><br>
        <li>Everyone needs to bring a towel and full water bottle.</li>
    </ol>
</div>
<div style="margin-bottom: 25px ; margin-right: 20px; margin-top: 10px; margin-left:
20px;" class="input-group">
    <h5><b>PERSONAL CONDUCT :</b></h5><br>
    <ol>
        <li>Phoenix members, guests and court rental guests (herein referred to as 'players)
are subject to the rules and guidance of the Phoenix staff and must wholly respect and follow staffs
instructions and club regulations at all times while playing in the club.</li><br>
        <li>Players agree to conduct themselves in a quiet and well-mannered fashion when in
the club and not cause any disturbances or interfere with the safe use or enjoyment of the academy by other
players.</li><br>
        <li>Members and Guests shall restrain themselves from using profanities while in the
premises.</li><br>
        <li>Members and Guests shall restrain from physical contact with another person in
any angry or threatening way. Harassment or intimidation by words, gestures. body language or any other menacing
behaviour.</li><br>
    </ol>
</div>
<div style="margin-bottom: 25px ; margin-right: 20px; margin-top: 10px; margin-left:
20px;" class="input-group">
    <input type="checkbox"> I hereby agree to abide by the terms and conditions as
provided by the Phoenix Board of Badminton Council.
</div>
<div style="margin-bottom: 25px; text-align: center;" class="input-register">

        <a routerLink="/booked" class= "btn btn-warning" style="color:black"
role="button">Book Court</a>
    </div>

```

- **USER MANAGEMENT COMPONENT:**

HTML:

```

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

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

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

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

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

```

```

        <br><br>

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

<div class="col" style="color: black;">
    <p style="color:black; text-align: center;">
        {{msg}}
    </p>
    <table class="table table-striped" style="background-color: blanchedalmond;">
        <thead>
            <tr>
                <th scope="col">S.No</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.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" style="color: snow; text-align: center; background-color: teal;">
        No Records Found
    </p>
</div>

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

```

TYPESCRIPT:

```

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 user?"))
    {
        //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 wrong";
        })
    }
}
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 style="text-align: center;">
                    {{msg}}
                </p>
            </div>
        </div>
    </div>

```

```

        <form #f="ngForm" (submit)="addUser(f)" 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"
(blur)="usernameCheck(username.value)" placeholder="Username" required minlength="6">
            </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 #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 #userpassword="ngModel"
placeholder="Password" minlength="6" required>
            </div>

            <p class="alert alert-danger"
*ngIf="userpassword.errors?.required && userpassword.touched">
                Please enter your password
            </p>
            <p class="alert alert-danger"
*ngIf="userpassword.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-
phone"></i></span>
                <input type="text" class="form-control" name="uphone" ngModel placeholder="Phone Number"
#userphone="ngModel" minlength="10" pattern="^(0|[1-9][0-9]*)$">
            </div>
            <p class="alert alert-danger"
*ngIf="userphone.errors?.pattern">

                Please Enter Nos Only
            </p>
            <p class="alert alert-danger"
*ngIf="userphone.errors?.minlength">
                Please 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="Add User" [disabled]="f.invalid ||
!usernameAvail" /> &nbsp;

            </div>
        </div>
    </form>
    <br> <br><br>

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

```

TYPESCRIPT:

```

import { Component, OnInit } from '@angular/core';
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(frmVal: any)
  {
    this.userSer.doUserRegistration(frmVal.value).subscribe((data:string)=>{

      console.log(data);

      this.msg = data;

      frmVal.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 style="text-align: center;">
                    {{msg}}
                </p>

                <form #f="ngForm" (submit)="editUser(f)" *ngIf="userdata" 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]="userdata.uname" #username="ngModel"
placeholder="Username" required minlength="6">
                    </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>

```

TYPESCRIPT:

```

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";
    })
  }
}

```

- **LOGOUT 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">Thank You!</div>
      </div>
      <div style="padding-top:30px" class="panel-body" >

        Your booking is confirmed!!
      </div>
    </div>
  </div>
</div>

```

```

        </div>
        <div style="text-align: center; margin-top: 50px; margin-left: 25px; padding-bottom: 30px;"
class="input-group-addfb">
            <a (click)="doLogOut()" class="btn btn-danger" style="color:whitesmoke"
role="button">Logout</a>

        </div>

    </div>

</div>
</div>

```

TYPESCRIPT:

```

import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

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

  constructor(public myRouter : Router) { }

  ngOnInit(): void {
  }

  doLogOut()
  {
    localStorage.clear();
    this.myRouter.navigateByUrl("/");
  }

}

```

- **FOOTER COMPONENT:**

HTML:

```

<div class="footer" style="margin: auto;">
    <p style="text-align: center;">Copyright 2021 | All Rights Reserved.</p>
</div>

```

- **USER SERVICE COMPONENT:**

```

import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'

```

```

})
export class UsersService {

  constructor(public http : HttpClient) { }

  doUserRegistration(data: any)
  {
    return this.http.post<string>("http://localhost:3000/register", data);
  }

  doUserLogin(data : any)
  {
    return this.http.post<any[]>("http://localhost:3000/login", data);
  }

  isLoggedIn()
  {
    return !!localStorage.getItem("loggeduser");
  }

  getAllUsers()
  {
    return this.http.get<any[]>("http://localhost:3000/allusers");
  }

  usernameCheckAvailability(username:string)
  {
    return this.http.get<any[]>("http://localhost:3000/usernamecheck/"+username);
  }

  getSingleUserData(userid : string)
  {
    return this.http.get<any[]>("http://localhost:3000/getuser/"+userid);
  }

  editSingleUserData(data: any)
  {
    return this.http.put<string>("http://localhost:3000/updateuser", data);
  }

  deleteUserData(userid: number)
  {
    return this.http.delete<string>("http://localhost:3000/deleteuser/"+userid);
  }

  searchUsers(searchtxt:string)
  {
    return this.http.get<any[]>("http://localhost:3000/searchuser/"+searchtxt);
  }

}

```

- **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 {
  console.log(this.userSer.isLoggedIn());

  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 { AdduserComponent } from './adduser/adduser.component';
import { AuthGuard } from './auth.guard';
import { BookedComponent } from './booked/booked.component';
import { CourtComponent } from './court/court.component';
import { EdituserComponent } from './edituser/edituser.component';
import { LoginComponent } from './login/login.component';
import { MembershipComponent } from './membership/membership.component';
import { RegisterComponent } from './register/register.component';
import { TermsComponent } from './terms/terms.component';
import { UsersComponent } from './users/users.component';

const routes: Routes = [
  {path: "register", component: RegisterComponent},
  {path: "", component: LoginComponent},
  {path: "court", component: CourtComponent, canActivate: [AuthGuard] },
  {path: "booked", component: BookedComponent, canActivate: [AuthGuard]},
  {path: "member", component: MembershipComponent, canActivate: [AuthGuard]},
  {path: "terms", component: TermsComponent, canActivate: [AuthGuard]},
  {path: "users", component: UsersComponent, canActivate: [AuthGuard]},
  {path: "adduser", component: AdduserComponent, canActivate: [AuthGuard]},
  {path: "edituser/:userid", component: EdituserComponent, canActivate: [AuthGuard]}
];

@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.end("Welcome to Phoenix Badminton court");

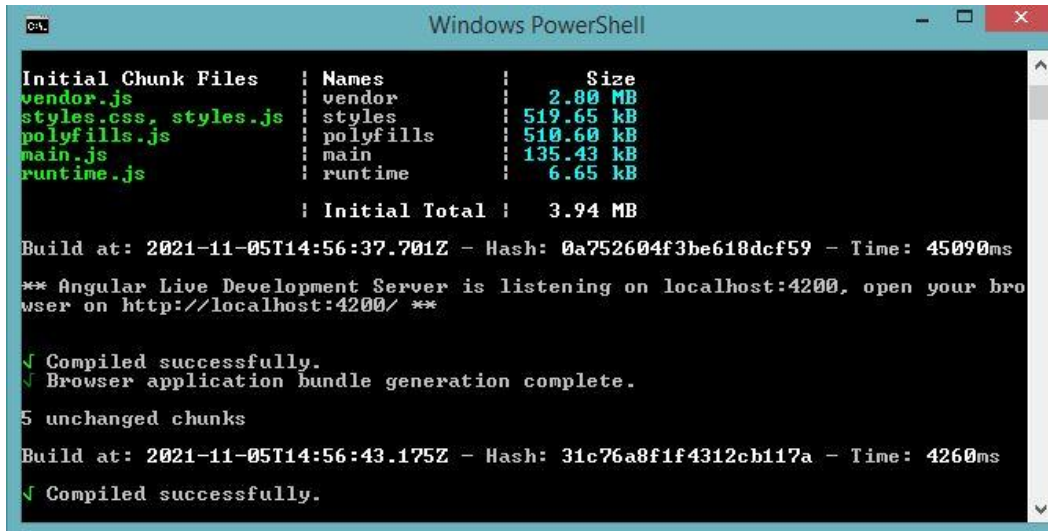
```

```
    //});  
    const server = http.createServer(app);  
  
    server.listen(3000, ()=>{  
        console.log("Check port 3000");  
    });
```

- **BASIC LAYOUT COMPONENT:**

```
<!doctype html>  
<html lang="en">  
<head>  
  <meta charset="utf-8">  
  <title>BadmintonCourtBooking</title>  
  <base href="/">  
  <meta name="viewport" content="width=device-width, initial-scale=1">  
  <link rel="icon" type="image/x-icon" href="favicon.ico">  
  <style>  
    .test{  
      width: 100%;  
      background-image: url(assets/images/background.jpg);  
  
    }  
  </style>  
</head>  
<body class="test">  
  <app-root></app-root>  
</body>  
</html>
```

B. SCREENSHOTS



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The terminal displays the output of an Angular build process. It lists the initial chunk files and their sizes, followed by the build time and hash. The output indicates that the application bundle generation is complete and that 5 unchanged chunks were found. The build process is shown to have completed successfully.

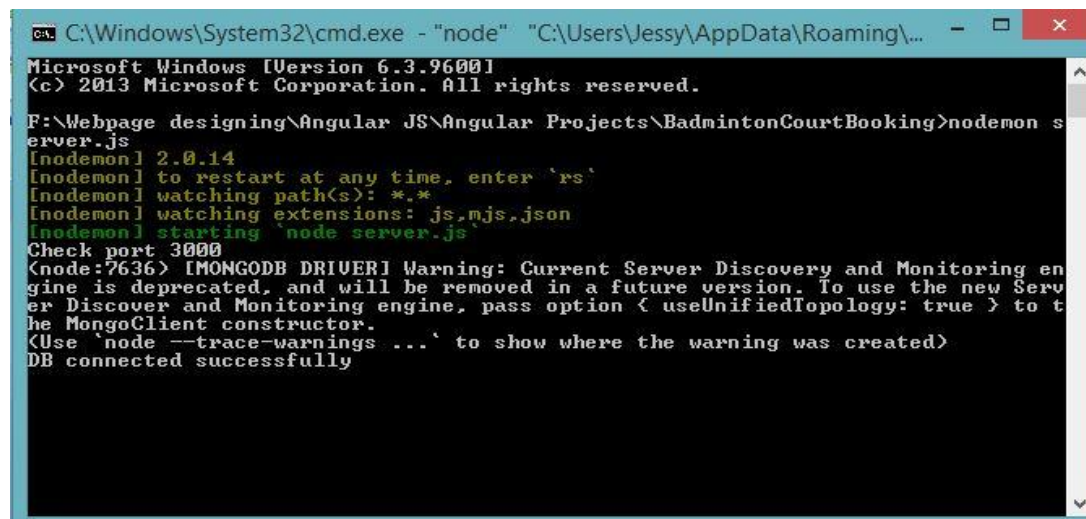
```
Initial Chunk Files | Names | Size
vendor.js          | vendor | 2.80 MB
styles.css, styles.js | styles | 519.65 kB
polyfills.js       | polyfills | 510.60 kB
main.js            | main | 135.43 kB
runtime.js         | runtime | 6.65 kB
                  | Initial Total | 3.94 MB

Build at: 2021-11-05T14:56:37.701Z - Hash: 0a752604f3be618dcf59 - Time: 45090ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
✓ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2021-11-05T14:56:43.175Z - Hash: 31c76a8f1f4312cb117a - Time: 4260ms
✓ Compiled successfully.
```



A screenshot of a Windows Command Prompt window titled "C:\Windows\System32\cmd.exe - 'node' 'C:\Users\Jessy\AppData\Roaming\...". The terminal shows the output of running the command "nodemon server.js". It displays the Nodemon version (2.0.14), the path being watched (*.*), and the extensions being watched (*.js, *.mjs, *.json). The output also shows the Node.js version (v16.13.0) and the MongoDB driver version (3.6.10). A warning message from the MongoDB driver is displayed, indicating that the current Server Discovery and Monitoring engine is deprecated and will be removed in a future version. The output concludes with "DB connected successfully".

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

F:\Webpage designing\Angular JS\Angular Projects\BadmintonCourtBooking>nodemon server.js
[nodemon] 2.0.14
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting 'node server.js'
Check port 3000
(node:7636) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.
(Use 'node --trace-warnings ...' to show where the warning was created)
DB connected successfully
```