LEARNING MANAGEMENT SYSTEM

A PROJECT COMPONENT REPORT

Submitted by

MUTHUKUMARASAMY S (Reg. No. 202104088)

RAGAVAMATHAVAN T (Reg. No. 202104110)

for the Theory Cum Project Component

of

19CS694 – WEB USER INTERFACE DESIGN

during

VI Semester – 2023 – 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

April 2024

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

(An Autonomous Institution affiliated to Anna University Chennai)

Department of Computer Science and Engineering

BONAFIDE CERTIFICATE

Certified that this project component report titled **LEARNING MANAGEMENT SYSTEM** is the bonafide work of **S.MUTHUKUMARASAMY(Reg. No. 202104088),** and **T.RAGAVAMATHAVAN(Reg. No. 202104110)** who carried out this work under my guidance for the Theory cum Project Component course **"19CS694 – WEB USER INTERFACE DESIGN"** during the sixth semester.

Dr. K. MUTHAMIL SUDAR, M.E.,Ph.D.
Assistant Professor
Course Instructor
Department of Computer Science &Engg.
Mepco Schlenk Engineering College
Sivakasi.

Dr. J. RAJA SEKAR, M.E.,Ph.D.
Professor
Head of the Department
Department of Computer Science &Engg.
Mepco Schlenk Engineering College
Sivakasi.

Submitted for viva-Voce Examination held at **MEPCO SCHLENK ENGINEERING COLLEGE (Autonomous), SIVAKASI** on **……/……/…. 20.....**

Internal Examiner **External Examiner**

ABSTRACT

The objective of implementing a Learning Management System (LMS) is to revolutionize the way educational content is delivered and managed. The LMS serves as a centralized platform that facilitates the creation, distribution, and tracking of learning materials, making education accessible anytime and anywhere. One of the distinctive features of the LMS is its ability to accommodate diverse learning styles and preferences, catering to both traditional classroom settings and modern online learning environments.

The significance of an LMS lies in its capacity to streamline the learning process for students, educators, and administrators alike. Students benefit from personalized learning paths, interactive content, and self-paced study options, fostering a more engaging and effective learning experience. Educators can leverage the LMS to design dynamic courses, monitor student progress, and provide timely feedback, enhancing their teaching capabilities and student outcomes. Administrators gain insights into learning analytics, resource utilization, and overall course effectiveness, enabling data-driven decision-making and continuous improvement in educational practices.

ACKNOWLEDGEMENT

First and foremost, we thank the **LORD ALMIGHTY** for his abundant blessings that is showered upon our past, present and future successful endeavors.

We extend our sincere gratitude to our college management and Principal **Dr.S. Arivazhagan M.E., Ph.D.,** for providing sufficient working environment such as systems and library facilities. We also thank him very much for providing us with adequate lab facilities, which enable us to complete our project.

We would like to extend our heartfelt gratitude to **Dr.J. Raja Sekar M.E., Ph.D.,** Professor and Head, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for giving me the golden opportunity to undertake a project of this nature and for his most valuable guidance given at every phase of our work.

We would also like to extend our gratitude and sincere thanks to **Dr.K. Muthamil Sudar M.E., Ph.D.,** Assistant Professor, Department of Computer Science and Engineering, Mepco Schlenk Engineering College for being our Project Mentor. He has put his valuable experience and expertise in directing, suggesting and supporting us throughout the Project to bring out the best.

Our sincere thanks to our revered **faculty members and lab technicians** for their help over this project work.

Last but not least, we extend our indebtedness towards out beloved family and our friends for their support which made the project a successful one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PERSCPECTIVE

The objective of implementing a Learning Management System (LMS) is to revolutionize the way educational content is delivered and managed. The LMS serves as a centralized platform that facilitates the creation, distribution, and tracking of learning materials, making education accessible anytime and anywhere. One of the distinctive features of the LMS is its ability to accommodate diverse learning styles and preferences, catering to both traditional classroom settings and modern online learning environments.

The significance of an LMS lies in its capacity to streamline the learning process for students, educators, and administrators alike. Students benefit from personalized learning paths, interactive content, and self-paced study options, fostering a more engaging and effective learning experience. Educators can leverage the LMS to design dynamic courses, monitor student progress, and provide timely feedback, enhancing their teaching capabilities and student outcomes. Administrators gain insights into learning analytics, resource utilization, and overall course effectiveness, enabling data-driven decision-making and continuous improvement in educational practices.

1.2 OBJECTIVES

The main objective is to provide accessible education services to underserved populations while reducing administrative burdens. This system enables easy course enrollment for students and efficient management tools for educators.

1.3 SCOPE

The scope of the project is to provide online learning services for remote learners and facilitate course management for educators within the Learning Management System.This scope outlines the specific objectives of the project related to online learning delivery and course management functionalities within the LMS.

CHAPTER 2

REQUIREMENT DESCRIPTION

2.1 FUNCTIONAL REQUIREMENTS

The functional requirements of the Learning Management System (LMS) encompass the following operations:

- **User Authentication:** The system should provide authentication functionality for valid users, including students, educators, and administrators, ensuring secure access to the platform. It should provide the functionality of real time notification sent to the receiver if the donor is ready to donate.
- **Request Handling:** The system should enable users, such as students, to submit requests for course enrollment, assessment submissions, or administrative support, with functionality for the system to accept or reject these requests based on predefined criteria.
- **Navigation Assistance:** The LMS should provide navigation functionality, allowing users to easily navigate within the platform, access course materials, participate in discussions, and track their progress.

2.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements of the Learning Management System (LMS) include:

- **Scalability:** The system should be scalable to accommodate a large number of users simultaneously accessing the platform without performance degradation. The application must be user friendly and must be very interactive to the user.
- **Maintenance and Support:** The system should have provisions for regular maintenance, updates, and technical support to address issues promptly and keep the platform operational and up-to-date.

CHAPTER 3

SYSTEM DESIGN

3.1 ARCHITECTURE DESIGN

The architecture of the Learning Management System (LMS) begins with user authentication. Users with accounts proceed to the login page, where successful login grants access to course enrollment, learning materials, communication tools, and administrative functionalities. Users without accounts are prompted to register before accessing the system's features. The LMS architecture ensures a seamless flow for users to engage in learning activities,interact with peers and instructors, and manage course-related tasks efficiently.

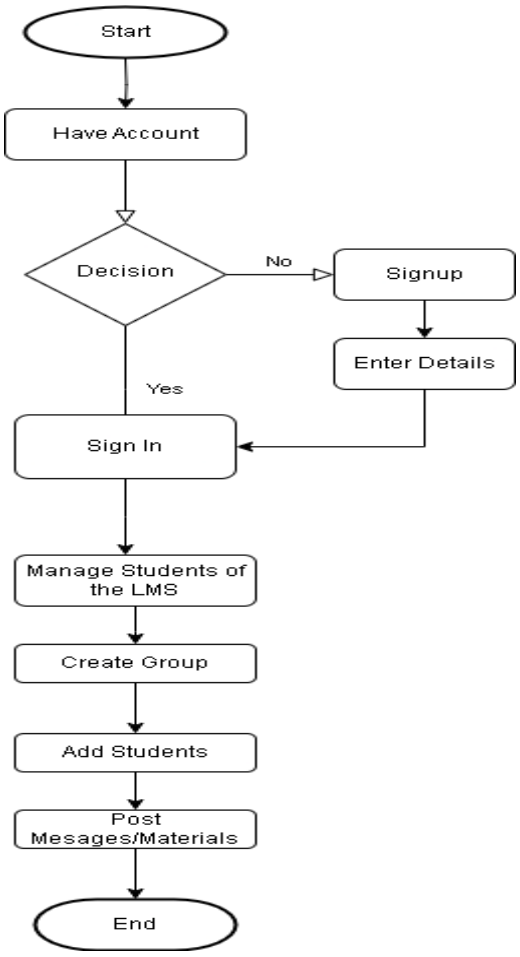

Figure 3.1: Architecture Diagram of Learning Management System

3.2 DESIGN COMPONENTS

3.2.1 Front End:

The learning Management System uses Angular and express Js for developing interactive pages.

3.2.2 Back End:

Uses Mongo database and nodes Js for backend to store data.

3.3 DATABASE DESCRIPTION

Listed below gives a description of database document schemas used for learning Management System.

3.3.1 User Structure

As shown in **Table 3.1**, user structure contains the details of the users, this Table is used to store the user information like name, email id, and password. This is the main user collection Table which is connected it signup and login page of the user to the web application.

Table 3.1: User Description

Attribute Name	Type	Constraint(s)	Description
Name	String	Not null	Name of the user
Email id	String	Should contain @symbol	Email id of the user
Password	String	Not null and minimum 8 characters	Password for the sign up
Ph No	String	Not null	Phone Number of the user

3.3.2 Staff Structure

As shown in **Table 3.2**, Staff structure contains the details of the Staff. In this collection the details of the Staff are stored such as name, email, role, contact details of the employee.

Table 3.2: Staff Description

Attribute Name	Type	Constraint(s)	Description
name	String	Not null	Name of the staff
email	String	Should contain @symbol	Staff email id
password	String	Not null with minimum 8 characters	password of the Staff
Ph no	integer	Maximum length of 10	Phone number of the Staff

3.3.3 Message Structure

As shown in **Table 3.3**, message structure contains the details of the Message. In this collection the details of the message are retrieved from the staff with the fields staff name, message, user name and timestamp.

Table 3.3: Message Description

Attribute Name	Type	Constraint(s)	Description
Staff name	String	Not null	Name of the Staff
Timestamp	integer	Follow the format dd/mm/yyyy hh:mm:ss	Time and date of Message
User name	String	Not null	Name of the User
Message	String	Not null	Message for the user

3.3.4 Group Structure

As shown in **Table 3.4**, group structure contains the details of the group. In this collection the details the group is provided by the staff who creates the group.

Table 3.4: Group Description

Attribute Name	Type	Constraint(s)	Description
Email	String	Should contain @symbol	Email of the staff who created the group
Gname	String	Not null	Name of the group

3.3.5 File Structure

As shown in **Table 3.5**, File structure contains the details of the File. In this collection the staff will upload the files to the groups where , the students can download it from their groups.

Table 3.5: File Description

Attribute Name	Type	Constraint(s)	Description
length	Integer	Not null	Length of the uploaded file
chunkSize	Integer	Length of the file uploaded	Chunksize of the uploaded file
uploadDate	Integer	It will be in the format yyyy/mm/dd	Date of the file when it is uploaded
filename	Integer	Not null	Name of the file uploaded

3.4 LOW LEVEL DESIGN

The following section illustrates the functionalities of the system. This includes login to the application, creating classes.

3.4.1 Signup

Table 3.6 shows the signup details of the user.

Table 3.6 signup Details

Files used	signup.component.html, signup.component.css, signup.component.ts
Short Description	Allows the user to register to the application
Arguments	Name, Email id, Password, Ph No
Return	Success/Failure in registering
Pre-Condition	The user must have an email account
Post-Condition	The home page will be displayed
Exception	Invalid email id and password
Actor	Staff , Student

3.4.2 Login

Table 3.7 shows the login details of the application.

Table 3.7 Login Details

Files used	Login.component.html, login.component.css, login.component.ts
Short Description	Allows the user to login to the application
Arguments	Email id, Password
Return	Success/Failure in login
Pre-Condition	The user must have an account
Post-Condition	The home page will be displayed
Exception	Invalid email id and password
Actor	Staff, Student

3.4.3 Staff login

Table 3.8 shows the admin login details of the application.

Table 3.8 Admin login Details

Files used	staff.component.html, staff.component.css, staff.component.ts
Short Description	Allows the Staff to login to the application
Arguments	Email id, Password
Return	Success/Failure in login
Pre-Condition	The Staff must have an account
Post-Condition	The home page will be displayed
Exception	Invalid email id and password
Actor	Staff

3.4.4 Messages

Table 3.9 shows the message sended by the staffs.

Table 3.9 Message Details

Files used	msg.component.html, msg.component.css, msg.component.ts
Short Description	Allows the staff to send messages to the stuents
Arguments	Email , Date , Time
Return	Success/Failure in sending message
Pre-Condition	The user should be aware of website
Post-Condition	The home page will be displayed
Exception	Nill
Actor	Staff

3.4.5 Group

Table 3.10 shows the created group for the students dashboard.

Table 3.10 Group Details

Files used	group.component.html,group.component.css,group.component.ts
Short Description	Allows the staffs to staffs to create various group by adding students
Arguments	Group name, staff email, message
Return	Success/Failure in contact submission
Pre-Condition	The user should be aware of website
Post-Condition	The home page will be displayed
Exception	Nil
Actor	Staff

3.5 USER INTERFACE DESIGN

3.5.1 Home

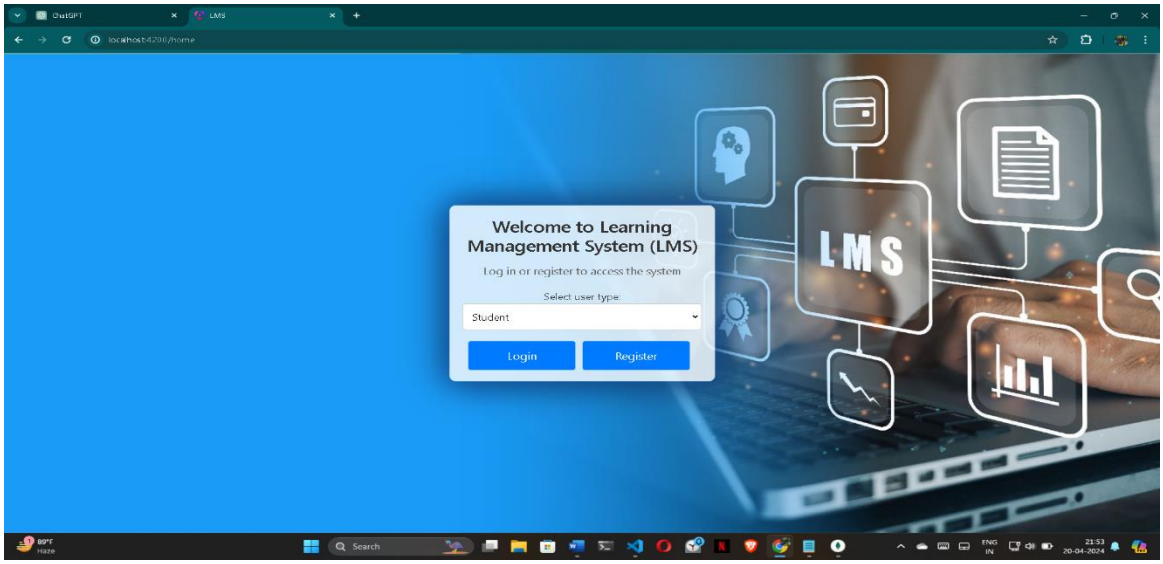

Figure 3.2: Main layout of learning Management System

This **Figure3.2** provides an interface for main activity of the learning management system where both staff and student can use the interface.

3.5.2 Login page

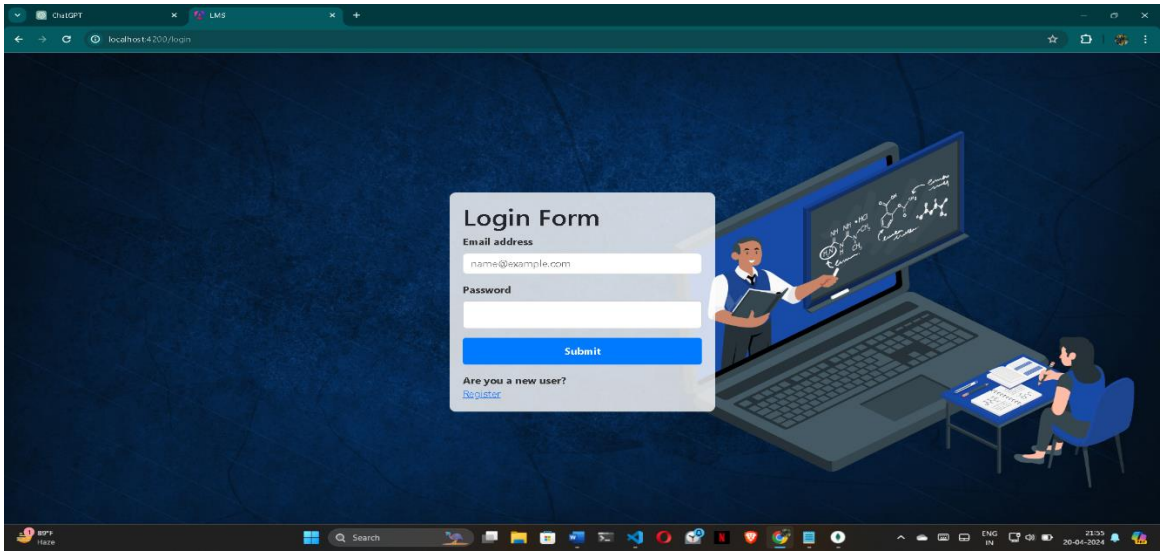

Figure 3.3: Login Page

This **Figure 3.3** provides interface about login page for registered user for the management system the user will be logged in with email id and password.

3.5.3 Signup page

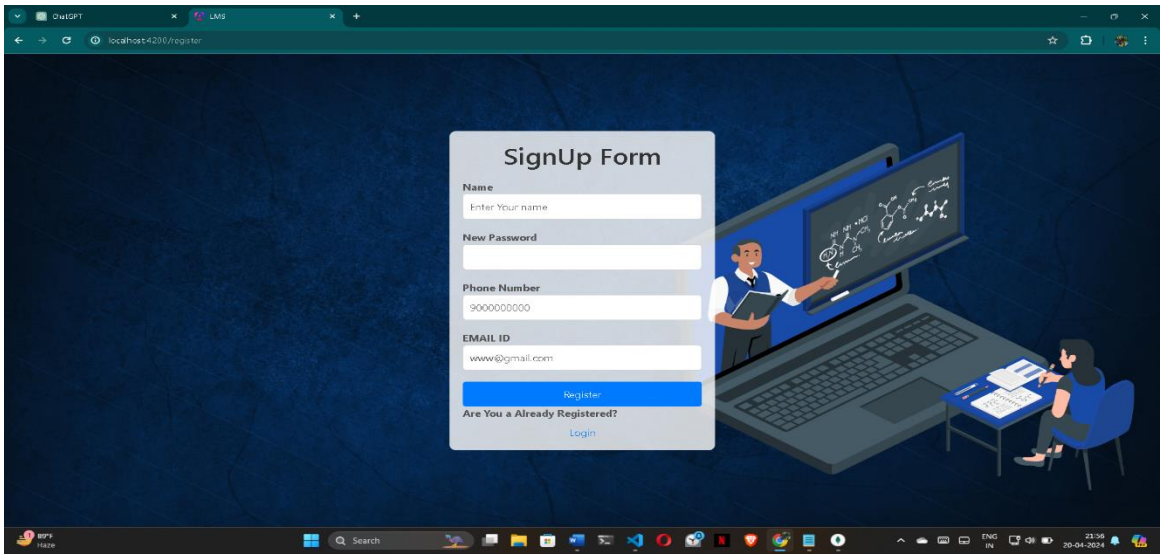

Figure 3.4: Signup Page

This **Figure 3.4,** this interface provides signup page for anonymous user, where the user can register with user name, email, password and phone number.

3.5.4 Student Home page

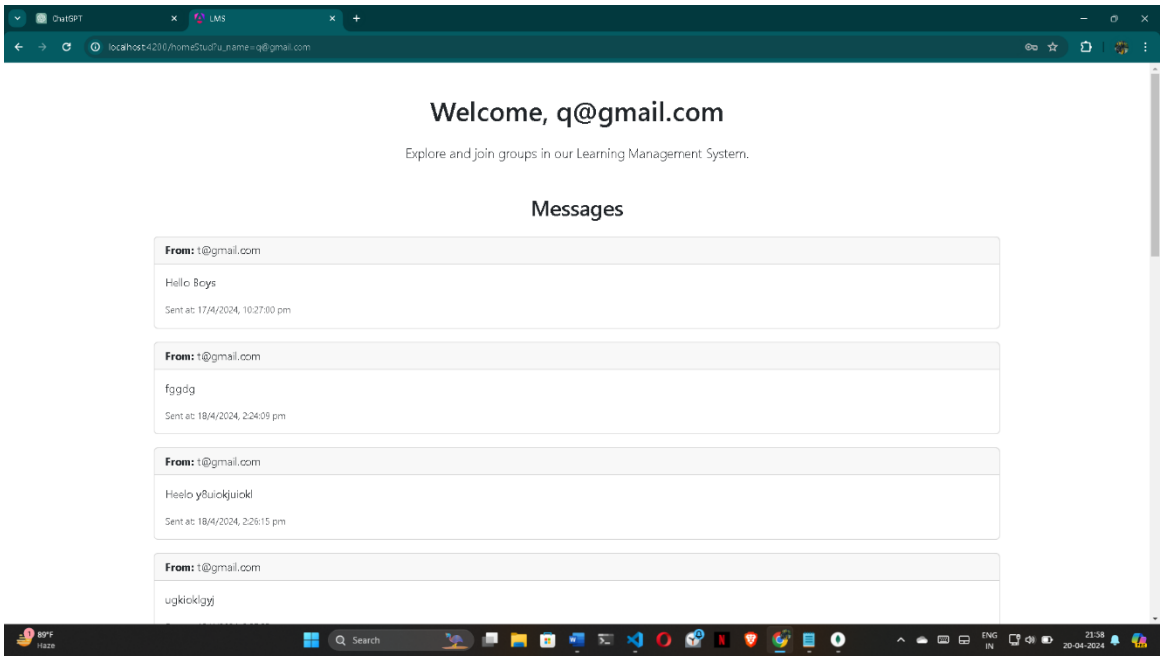

Figure 3.5: Student Home page

This **Figure 3.5,** provides interface for student page where student can be able to view the message sended by the staff in their respective groups.

3.5.5 Staff home page

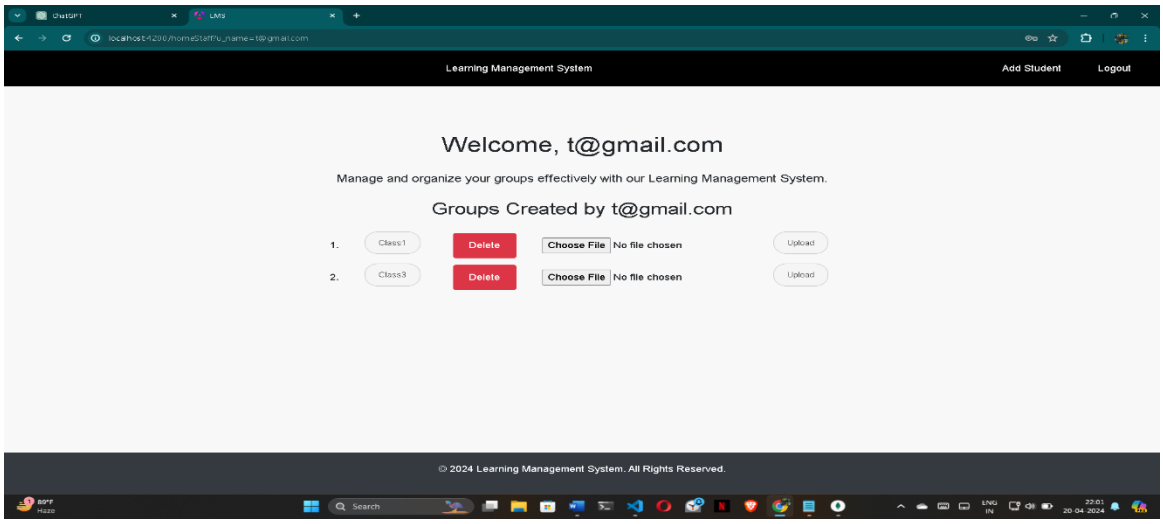

Figure 3.6: Staff home page

This **Figure 3.6,** provides the interface for staff ,where the staff can be able to add student , create group and delete group.

3.5.6 Group creation page

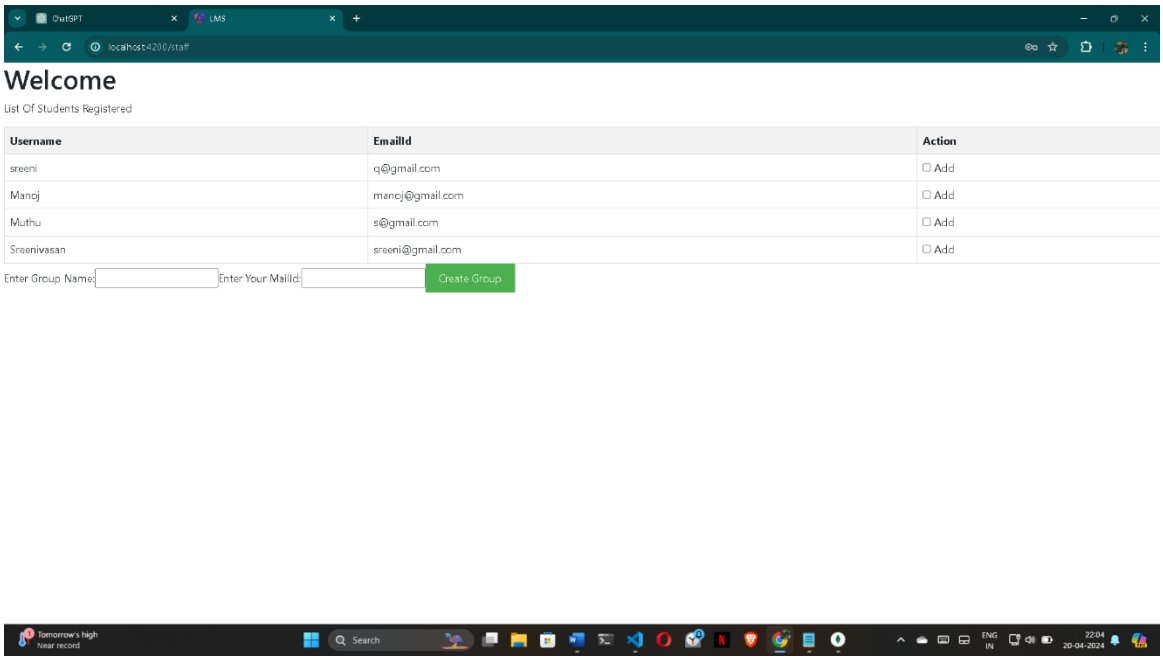

Figure 3.7: Group creation page

This **Figure 3.7,** provides the interface for creating group by viewing list of students and adding them to the group by entering group name and mail id of the staff.

3.5.7 Quiz creation page

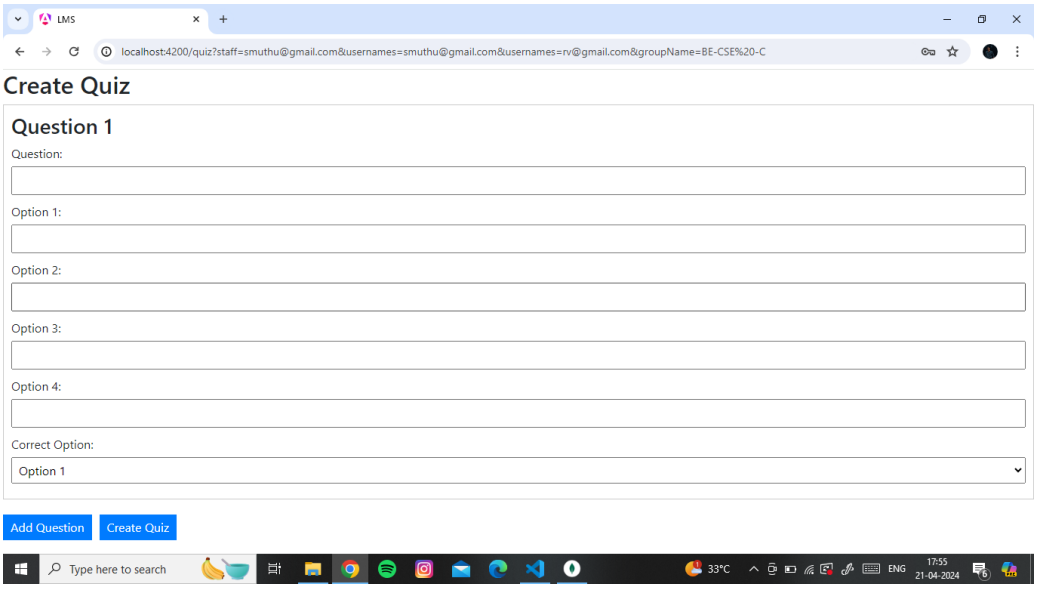

Figure 3.8: Quiz creation page

This **Figure 3.8,** provides the interface for creating quiz by providing list of questions and adding them to the DB by entering group name and mail id of the staff.

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 LOGIN IMPLEMENTATION

The user logs in to the application with the registered email id to the learning management system. The user of the application is authenticated and validate with the data provide by him to the login credential, once the user is logged in to the website, he is navigated to the user login where the user can view the messages in the group and download the attachments.

```
GET email, password
   IF userid, password valid
         RETURN homepage
   ELSE
         TOAST Invalid Credential
```

4.2 SIGNUP IMPLEMENTATION

The signup form is to register the user with the application, once the user is registered to the application with the email id they can be able to login into the application.

```
GET requestFields
   IF requestFields valid
         RETURN added to db
   ELSE
         TOAST enter valid details
```

4.3 STAFF LOGIN IMPLEMENTATION

the user logs in to the application with the registered email id to the learning management system. The staff of the application is authenticated and validate with the data provide by him to the login credential, once the staff is logged in to the website he is navigated to the user login where he can create, update and deletegroups.

```
GET email, password
```

```
    IF userid, password valid
        RETURN homepage
    ELSE
        TOAST Invalid Credential
```

4.4 MESSAGE POSTING IMPLEMENTATION

The message page in the application is to provide the message sended by the staffto the groups along with some attachments like pdf and documents.

```
GET requestFields
    IF requestFields valid
        RETURN added to db
    ELSE
        TOAST enter valid details
```

4.5 ADD STUDENT IMPLEMENTATION

This page provides functionality to add students to the group by the staffs.

4.6 CREATE GROUP IMPLEMENTATION

This page is used to create new groups in the learning management system, by the staffs.

```
GET requestFields
    IF requestFields valid
        RETURN added to db
    ELSE
        TOAST enter valid details
```

4.7 DELETE GROUP IMPLEMENTATION

This page is used to delete the groups in the learning management system, by the staff of the application.

```
GET requestFields
    IF requestFields valid
        RETURN update or delete the details
    ELSE
        TOAST enter valid details
```

CHAPTER 5

RESULTS AND DISCUSSION

5.1 TEST CASES AND RESULTS

5.1.1 Test Cases and Results for Login function:

The **Table 5.1, Table 5.2** shows that the possible test data for the both positive and negative test case given below, if the user is already having account, then the output is true otherwise false.

Table 5.1:Positive Test Case and result for Login

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	muthu@gmail.com,1234
Expected Output	TRUE
Result	PASS

Table 5.2: Negative Test Case and result for Login

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	9490718418
Expected Output	FALSE
Result	PASS

5.1.2 Test Cases and Results for Signup function:

The **Table 5.3, Table 5.4** shows that the possible test data for the both positive and negative test case given below, if the user is registered with valid details, then return true else return false.

Table 5.3:Positive Test Case and result for signup

Test Case ID	TC1
Test Case Description	It tests whether the given signup details are valid or not
Test Data	muthu, muthu@gmail.com, 1234,982374651
Expected Output	TRUE
Result	PASS

Table 5.4: Negative Test Case and result for signup

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	9490718418
Expected Output	FALSE
Result	PASS

5.1.3 Test Cases and Results for Staff Login function:

The **Table 5.5, Table 5.6** shows that the possible test data for the both positive and negative test case given below, if the staff is already having account, then the output is true otherwise false.

Table 5.5:Positive Test Case and result for StaffLogin

Test Case ID	TC1
Test Case Description	It tests whether the given adminlogin details are valid or not
Test Data	ragav@gmail.com, hardin123
Expected Output	TRUE
Result	PASS

Table 5.6: Negative Test Case and result for Staff Login

Test Case ID	TC2
Test Case Description	It tests whether the given adminlogin details are valid or not
Test Data	9490718418,2343
Expected Output	FALSE
Result	PASS

5.1.4 Test Cases and Results for Create Group function:

The **Table 5.7, Table 5.8** shows that the possible test data for the both positive and negative test case given below, if the Staff is already having account, then the output is true otherwise false.

Table 5.7:Positive Test Case and result for create group

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	Group CSE ,hardin@gmail.com
Expected Output	TRUE
Result	PASS

Table 5.8: Negative Test Case and result for create group

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	9490718418
Expected Output	FALSE
Result	PASS

5.1.5 Test Cases and Results for Delete group function:

The **Table 5.9, Table 5.10** shows that the possible test data for the both positive and negative test case given below, if the staff is already registered the staff can delete the group from the database, then the output is true otherwise false.

Table 5.9:Positive Test Case and result for edit group

Test Case ID	TC1
Test Case Description	It tests whether the given login details are valid or not
Test Data	muthu@gmail.com ,1234
Expected Output	TRUE
Result	PASS

Table 5.10: Negative Test Case and result for edit group

Test Case ID	TC2
Test Case Description	It tests whether the given login details are valid or not
Test Data	9490718418
Expected Output	FALSE
Result	PASS

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT(S)

In conclusion, our Learning Management System (LMS) project has laid the foundation for a user-friendly platform that facilitates online learning activities for students, educators, and administrators. Moving forward, we envision several enhancements to further improve the system's functionality and user experience.One key area for enhancement is advanced user tracking through analytics tools, enabling personalized learning support and course recommendations based on individual progress and learning patterns. Integration of multimedia content, such as videos, simulations, and interactive exercises, will enhance the learning experience and engagement levels.

Collaborative learning features, including group projects, peer reviews, and virtual study groups, will promote teamwork, communication, and knowledge sharing among students. Mobile compatibility is another essential enhancement, ensuring that users can access course materials and participate in activities seamlessly on mobile devices.Gamification elements, such as quizzes, badges, and leaderboards, will be incorporated to increase motivation and reward student achievements. Expanding the range of course offerings to cover diverse subjects and interests will cater to the evolving educational landscape.

Overall, these future enhancements will make our LMS more robust, user-friendly, and adaptable to the evolving needs of learners and educators in the digital age.

APPENDIX – A

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENT:

Memory	:	128GB RAM
Processor	:	Core i5 Processor
Disk	:	40GBOperating

SOFTWARE REQUIREMENT:

Operating System	:	Windows 11 (64 bit)
DBMS	:	MongoDB
IDE used	:	Visual Studio Code
Markup language	:	HTML
Programming language	:	Typescript
Scripting language	:	CSS
Framework	:	Angular

APPENDIX – B

SOURCE CODE

App.route.ts:

```
import { Routes } from '@angular/router';
import { RegisterComponent } from './register/register.component';
import { LoginComponent } from './login/login.component';
import { HomeComponent } from './home/home.component';
import { HomeStudComponent } from './home-stud/home-stud.component';
import { RegStaffComponent } from './reg-staff/reg-staff.component';
import { LogStaffComponent } from './log-staff/log-staff.component';
import { StaffComponent } from './staff/staff.component';
import { HomeStaffComponent } from './home-staff/home-staff.component';
import { PostmsgComponent } from './postmsg/postmsg.component';
import { ProfileStaffComponent } from './profile-staff/profile-staff.component';
import { GroupDetComponent } from './group-det/group-det.component';
import { DelGroupComponent } from './del-group/del-group.component';
import { QuizCreateComponent } from './quiz-create/quiz-create.component';
import { AttendQuizComponent } from './attend-quiz/attend-quiz.component';
import { Staff1Component } from './staff1/staff1.component';
import { StafflistComponent } from './stafflist/stafflist.component';
import { AuthGuard } from './auth.guard';
//import { AuthGuard } from './auth.guard';

export const routes: Routes = [
        {path:'',redirectTo:'/home' ,pathMatch:'full'},
        {path:'home',component:HomeComponent},
        { path: 'register', component: RegisterComponent },
```

```
        { path: 'login', component: LoginComponent },
        {path:'homeStud',component:HomeStudComponent,canActivate: [AuthGuard] },
        {path:'reg-staff',component:RegStaffComponent},
        {path:'log-staff',component:LogStaffComponent},
        {path:'staff', component:StaffComponent,canActivate: [AuthGuard]},
        {path:'homeStaff',component:HomeStaffComponent ,canActivate: [AuthGuard]},
        {path:'postmsg',component:PostmsgComponent,canActivate: [AuthGuard]},
        {path:'profile-staff',component:ProfileStaffComponent,canActivate: [AuthGuard]},
        {path:'group-det',component:GroupDetComponent,canActivate: [AuthGuard]},
        {path:'del-grp',component:DelGroupComponent,canActivate: [AuthGuard]},
            {path:'quiz',component:QuizCreateComponent,canActivate: [AuthGuard]},
            {path:'attend',component:AttendQuizComponent,canActivate: [AuthGuard]},
            {path:'grouplist',component:Staff1Component,canActivate: [AuthGuard]},
            {path:'stafflist',component:StafflistComponent,canActivate: [AuthGuard]}
    ];
```

HomeStud.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { LoginService } from '../login.service';
import { ActivatedRoute } from '@angular/router';
import { Observable } from 'rxjs';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import saveAs from 'file-saver';
import { ApiService } from '../api.service';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router';
import { AuthService } from '../auth.service';
@Component({
```

```
  selector: 'app-home-stud',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './home-stud.component.html',
  styleUrl: './home-stud.component.css'
})
export class HomeStudComponent implements OnInit{
  messages: Message[] = [];
  constructor(private router:Router,private http:HttpClient,private loginService:
LoginService,private route:ActivatedRoute,private apiService:ApiService,private
authService:AuthService) {}
  u_name: string = '';
  hasQuizAssigned: boolean = false;
  userId: string='';
  filenames:string[] =[];
  groups:any[]=[];
  ngOnInit() {
    if (!this.authService.isAuthenticated()) {
      this.router.navigate(['/login']); // Redirect to login page
    }
    //this.fetchMessages();
    this.http.get<string[]>('http://localhost:4000/filenames')
      .subscribe(filenames => {
        this.filenames = filenames;
      });
    this.route.queryParams.subscribe(params => {
      this.u_name = params['u_name'];
      this.fetchMessages();


      this.getGroupsByStud(this.u_name).subscribe((groups) => {
```

```
      this.groups = groups;
    }, (error: any) => {
      console.error('Error fetching groups:', error);
});
  });
}
downloadFile(filename: string): void {
  this.http.get(`http://localhost:4000/download/${filename}`).subscribe(() => {
  alert(`Downloading file: ${filename}`);
});
}
attendQuiz(username: string): void {
  this.router.navigate(['/attend'], { queryParams: { username } });
}
navigateToAttendQuiz(groupName: string) {
  this.router.navigate(['/attend-quiz'], { queryParams: { groupName } });
}
navigateTo(route: string): void {
  this.router.navigateByUrl(route);
}
navigateTo1(route: string): void {
  this.router.navigateByUrl(route);
  this.authService.logout();
}
  getGroupsByStud(staffName: string): Observable<any[]> {
    const url = `http://localhost:4000/groups?u_name=${staffName}`;
    return this.http.get<any[]>(url);
  }
  getGroupsByUserId(userId: string): Observable<any[]> {
```

```
    const url = `http://localhost:4000/groups?userId=${userId}`;
    return this.http.get<any[]>(url);
  }
  fetchMessages() {
    console.log(this.u_name)
    this.apiService.getMessages(this.u_name).subscribe(
      (messages: any) => {
        this.messages = messages;


      },


      (error) => {
        console.error('Error fetching messages:', error);
      }
    );
  }

  downloadPdf(pdfId: string) {
    this.apiService.getPdf(pdfId).subscribe((data: Blob) => {
      const downloadURL = window.URL.createObjectURL(data);
      const link = document.createElement('a');
      link.href = downloadURL;
      link.download = 'file.pdf';
      link.click();
    });
  }

}
 interface Message {
```

```
  _id: string;
  staffName: string;
  usernames: string[];
  mesg: string;
  timestamp: string;
  gname:string;
}
```

HomeStaff.component.ts:

```
import { Component, ElementRef, OnInit, ViewChild } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { ApiService } from '../api.service';
import { UploadService } from '../upload.service';
import { FormBuilder, FormGroup, ReactiveFormsModule } from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Observable, Subscription } from 'rxjs';
import { CommonModule } from '@angular/common';
import * as CryptoJS from 'crypto-js';
import { AuthService } from '../auth.service';

@Component({
  selector: 'app-home-staff',
  standalone: true,
  imports: [ReactiveFormsModule,CommonModule],
  templateUrl: './home-staff.component.html',
  styleUrls: ['./home-staff.component.css']
})
export class HomeStaffComponent implements OnInit {
  uploadForm: FormGroup;
  pdfs:any;
```

```
message: string | undefined;
u_name: string = '';
groupName: string = '';
groups: any[] = [];
selectedFile: File | null = null;
@ViewChild('singleInput', { static: false })
singleInput!: ElementRef;
private routeSubscription: Subscription | null = null;
constructor(
  private http: HttpClient,
  private router: Router,
  private apiService: ApiService,
  private uploadService: UploadService,
  private formBuilder: FormBuilder,
  private route: ActivatedRoute,
  private authService:AuthService
) {
  this.uploadForm = this.formBuilder.group({
    file: ['']
  });
}

navigateTo(route: string) {

  this.router.navigateByUrl(route);
  this.authService.logout();



}
```

```
  navigateTo3(route: string) {
    this.router.navigateByUrl(route)
  }
  navigateTo1(route:string,u_name:string,groupName:string)
  {

    this.router.navigate([route], { queryParams: { u_name: u_name, groupName: groupName } });

  }
    deleteGroup(u_name: string, gname: string): void {
      this.apiService.deleteGroup(u_name, gname).subscribe(
        () => {
          console.log('Group deleted successfully.');
          alert('Group deleted successfully');
          this.getGroupsByStaff(u_name);

          window.location.reload();
        },
        (error) => {
          console.error('Failed to delete group:', error);
          // Handle the error (e.g., display an error message)
        }
      );
    }
  ngOnInit(): void {
    if (!this.authService.isAuthenticated()) {
      this.router.navigate(['/log-staff']); // Redirect to login page
    }
    this.routeSubscription=this.route.queryParams.subscribe(params => {
```

```
    this.u_name = params['u_name'];
    this.getGroupsByStaff(this.u_name).subscribe(groups => {
      this.groups = groups;
    }, error => {
      console.error('Error fetching groups:', error);
      // Handle the error (e.g., show a message to the user)
    });

  });
}
ngOnDestroy(): void {
  if (this.routeSubscription) {
    this.routeSubscription.unsubscribe();
  }
}
fetchGroups() {
  this.http.get<any[]>('http://localhost:4000/retrievegroups').subscribe(groups => {
    this.groups = groups;
  });
}
  getGroupsByStaff(staffName: string): Observable<any[]> {
  const url = `http://localhost:4000/groups?u_name=${staffName}`;
  return this.http.get<any[]>(url);
}
getGroupByGroupNameAndUName(groupName: string): void {
  console.log(groupName)
  this.apiService.getGroupByGroupNameAndUName(groupName, this.u_name)
    .subscribe((data: any) => {
      this.groups = data;      });
```

```
  }
  selectPdf(event: any) {
    const fileList: FileList = event.target.files;
    if (fileList.length > 0) {
      this.pdfs = fileList[0];
      console.log(this.pdfs);
    }
  }
  onSubmit(gname:string){
    if (!this.pdfs) {
      console.error('No file selected.');
      return;
    }
const formData=new FormData();
formData.append('file',this.pdfs)
console.log(gname);
formData.append('gname',gname);
console.log(formData.get('gname'))
  const url = `http://localhost:4000/upload`;
this.http.post<any>(url,formData).subscribe((res)=>{
  alert("File Uploaded Successfully")
  //this.singleInput.nativeElement.value=" ";
},
  err=>{
    console.log(err);
})

  }
}
```

Group-det.component.ts:

```
import { CommonModule } from '@angular/common';
import { HttpClient } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Route } from '@angular/router';
import { ApiService } from '../api.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-group-det',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './group-det.component.html',
  styleUrl: './group-det.component.css'
})
export class GroupDetComponent implements OnInit{
  groupMembers: string[] = [];
  myName:string='';
  groupName:string='';
  constructor(private apiService: ApiService, private route: ActivatedRoute,private
router:Router) {}
  uploadPdf(): void {
    this.router.navigateByUrl('/upload-pdf');
  }
  navigateToPostMes(staff: string, usernames: string[],groupName:string) {

this.router.navigate(['/postmsg'], { queryParams:{ staff, usernames, groupName }});

  }
  assignQuiz(staff: string, usernames: string[],groupName:string) {
```

```
    this.router.navigate(['/quiz'], { queryParams:{ staff, usernames, groupName }});


      }
  ngOnInit(): void {
      this.route.queryParams.subscribe(params => {
        const staffName = params['u_name'];
        const groupName = params['groupName'];
        this.groupName=params['groupName']
        this.myName=params['u_name'],
        console.log(staffName,groupName)
      this.apiService.getGroupMembers(staffName, groupName).subscribe(
        (groups: any[]) => {
          this.groupMembers = groups.flatMap(group => group.email);
        },
        (error) => {
          console.error(error);
          // Handle error
        }
      );
    });
  }
  }
```

staffGroup.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from '../api.service';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { HttpClient } from '@angular/common/http';
import { Router } from '@angular/router';
```

```
import { AuthService } from '../auth.service';
interface User {
  id: number;
  name: string;
  email: string;
  selected: boolean;
}
interface Group{
  groupName: string;
  staffName:string;
}

@Component({
  selector: 'app-staff',
  standalone: true,
  imports: [CommonModule,FormsModule],
  templateUrl: './staff.component.html',
  styleUrl: './staff.component.css'
})
export class StaffComponent implements OnInit {
  users: any[]=[];
  groups:any[]=[];
  constructor(private http:HttpClient,private userService: ApiService,private
router:Router,private authService:AuthService) {}
  selectedUsers: User[] = [];



  group: Group = {
    groupName: "",
    staffName:""
```

```
  };
  createGroup(): void {
    const selectedUsers = this.users.filter(user => user.selected);
    if(this.group.staffName.length == 0 ||this.group.groupName.length == 0 ||
selectedUsers.length == 0){
      alert("Enter Group name and select students");
      return;
    }
    const groupData = {
      staffName:this.group.staffName,
      groupName: this.group.groupName,
      users: selectedUsers.map(user => ({ email: user.email, username: user.name }))
    };
    console.log(groupData)
    this.userService.createGroup(groupData).subscribe(
      response => {
        console.log('Group created successfully:', response);
        alert("Group Created Successfully");


        this.router.navigate(['/homeStaff'], { queryParams:{u_name: this.group.staffName } 
});
        this.users.forEach(user => user.selected = false);
      },
      error => {
        console.error('Error creating group:', error);


      }
    );
  }
```

```
    ngOnInit(): void {
    this.userService.getUsers().subscribe(users => {
      this.users = users;
    });
  }
}
```

Create-quiz.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { ApiService } from '../api.service';
import { CommonModule } from '@angular/common';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { ActivatedRoute } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import {Router} from '@angular/router'
@Component({
  selector: 'app-quiz-create',
  standalone: true,
  imports: [CommonModule,ReactiveFormsModule,FormsModule],
  templateUrl: './quiz-create.component.html',
  styleUrl: './quiz-create.component.css'
})
export class QuizCreateComponent implements OnInit{
  usernames: string[] = [];
  groupname:string='';
  questions: { question: string, options: string[], correctOption: number }[] = [];
  question: string = '';
  options: string[] = ['', '', '', ''];
  correctOption: number = 0;
  u_name :string=''
```

```
 constructor(private quizService: ApiService,private route: ActivatedRoute,private
router:Router) {}

 ngOnInit(): void {

   this.route.queryParams.subscribe(params => {

     this.usernames = Array.isArray(params['usernames']) ? params['usernames'] :
[params['usernames']];

     this.u_name=params['staff']

   });

 }

 addQuestion() {

   this.questions.push({

     question: '',

     options: ['', '', '', ''],

     correctOption: 0

   });

 }


 onSubmit() {

   if (this.questions.length === 0) {

     alert('Please add at least one question before submitting.');

     return;

   }

   if (this.questions.some(question => question.question.trim() === '')) {

     alert('Please provide a question for all questions before submitting.');

     return;

   }

   if (this.questions.some(question => question.options.some(option => option.trim() ===
'''))) {

     alert('Please provide options for all questions before submitting.');

     return;
```

```
    }
    if (this.questions.some(question => question.correctOption === -1)) {
      alert('Please select a correct option for all questions before submitting.');
      return;
    }
    this.questions.forEach(question => {

      const quizData = {
        usernames: this.usernames,
        question: question.question,
        options: question.options,
        correctOption: question.options[question.correctOption]
      };
console.log(quizData);
      this.quizService.createQuiz(quizData).subscribe(
        (response) => {
          console.log(response);
        },
        (error) => {
          console.error(error);
          // Handle error
        }
      );
    });

    alert('Quiz Created Successfully')
    this.router.navigate(['/homeStaff'], { queryParams: { u_name: this.u_name } });

  }
```

```
}
```

Attend-quiz.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { ApiService } from '../api.service';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import {Router} from '@angular/router'
@Component({
  selector: 'app-attend-quiz',
  standalone:true,
  imports:[FormsModule,CommonModule],
  templateUrl: './attend-quiz.component.html',
  styleUrls: ['./attend-quiz.component.css']
})
export class AttendQuizComponent implements OnInit {
  questions: any[] = [];
  selectedAnswers: { [key: string]: string } = {};
  timeLimit: number = 600; // Time limit in seconds (e.g., 10 minutes)
  timer: any;
  timeLeft: number=0;
  username: string='';
  constructor(private route: ActivatedRoute,private quizService: ApiService,private
router: Router) { }


  ngOnInit(): void {
    this.fetchQuizData();
    this.route.queryParams.subscribe(params => {
      this.username = params['username'];
      this.loadAssignedQuizzes();
```

```
  });
}
loadAssignedQuizzes() {
  this.quizService.getAssignedQuizzesByUsername(this.username)
    .subscribe((quizzes: any[]) => {
      this.questions = quizzes;
      if (this.questions.length === 0) {
        alert('No quizzes assigned.');
        this.router.navigate(['/homeStud'], { queryParams: { u_name: this.username } });


      }
    });
}

fetchQuizData() {
  this.quizService.getQuizData1().subscribe(
    (data: any[]) => {
      this.questions = data;
      this.initializeSelectedAnswers();
      this.startTimer();
    },
    (error: any) => {
      console.error('Error fetching quiz data:', error);
    }
  );
}

initializeSelectedAnswers() {
  this.questions.forEach(question => {
```

```
    this.selectedAnswers[question._id] = '';
  });
}

selectAnswer(questionId: string, selectedOption: string) {
  this.selectedAnswers[questionId] = selectedOption;
}

startTimer() {
  this.timeLeft = this.timeLimit;
  this.timer = setInterval(() => {
    this.timeLeft--;
    if (this.timeLeft === 0) {
      clearInterval(this.timer);
      this.submitQuiz();
    }
  }, 1000);
}



  submitQuiz() {
    clearInterval(this.timer); // Stop the timer

    let totalMarks = 0;

    // Iterate through questions to compare answers
    this.questions.forEach(question => {
      const correctAnswer = question.correctOption;
      const userAnswer = this.selectedAnswers[question._id];
```

```
      // Check if user answer is correct
      if (userAnswer === correctAnswer) {
        totalMarks++;
      }
    });

    // Calculate percentage of correct answers
    const percentage = (totalMarks / this.questions.length) * 100;

    // Display marks as an alert
    alert(`You scored ${totalMarks} out of ${this.questions.length}. Percentage: ${percentage.toFixed(2)}%`);
    this.router.navigate(['/homeStud'], { queryParams: { u_name: this.username } });
  }

  formatTime(seconds: number): string {
  const minutes = Math.floor(seconds / 60);
  const remainingSeconds = seconds % 60;
  return `${minutes}:${remainingSeconds < 10 ? '0' : ''}${remainingSeconds}`;}}
```

REFERENCES

1. Node js:https://www.geeksforgeeks.org/how-to-build-hospital-management-system-using-node-js/
2. Crud operations using mean: https://www.dotnettricks.com/learn/angularmaterial/dataTable-crud-operations-mean-stack#:~:text=Let%20me%20explain%20that%20here,of%20employees%20in%20the%20Table.
3. Html and css: https://www.positronx.io/mean-stack-tutorial-angular-crud-bootstrap/