



ONLINE COURSE MANAGEMENT SYSTEM



U19CSPR601 MINI PROJECT

Submitted by

MIRUTHULA J	(714019104056)
MONICA S	(714019104058)
RAM HARIDHRA R	(714019104079)

**in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
IN**

COMPUTER SCIENCE AND ENGINEERING

**SRI SHAKTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS),
COIMBATORE 641 062**

**Autonomous Institution, Accredited by NAAC with “A” Grade
ANNA UNIVERSITY: CHENNAI 600 25**

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this Report titled “**ONLINE COURSE MANAGEMENT SYSTEM**” is the bonafide work of “MIRUTHULA J (714019104056), MONICA S (714019104058) and, RAM HARIDHRA R (714019104079)” who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. K. E. Kannammal

HEAD OF THE DEPARTMENT

Professor and Head,

Department of CSE,

Sri Shakthi Institute of Engineering

and Technology,

Coimbatore- 641 062.

SIGNATURE

Mrs M. BanuPriya

SUPERVISOR

Assistant Professor,

Department of CSE,

Sri Shakthi Institute of Engineering

and Technology,

Coimbatore- 641 062.

Submitted for the project work viva-voce Examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, I would like to thank God Almighty for giving me strength.

Without his blessings, this achievement would not have been possible.

We express our deepest gratitude to our **Chairman Dr. S. Thangavelu** for his continuous encouragement and support throughout our course of study.

We are thankful to our **Secretary Er .T. Dheepan** for his unwavering support during the entire course of this project work.

We are also thankful to our Joint **Secretary Mr. T. Sheelan** for his support during the entire course of this project work.

We are highly indebted to Principal **Dr. A. R. Ravi Kumar** for their support during the tenure of the project.

We are deeply indebted to our **Head of the Department**, Computer Science and Engineering, **Dr. K. E. Kannammal**, for providing us with the necessary facilities.

It's a great pleasure to thank our **Project Guide Mrs. M. Banupriya** for her valuable technical suggestions and continuous guidance throughout this project work.

We solemnly extend our thanks to all the teachers and non-teaching staff of our department, family, and friends for their valuable support.

MIRUTHULA J (714019104056)

MONICA S (714019104058)

RAM HARIDHRA R (714019104079)

ABSTRACT

Requirements definition and management is recognized as a necessary step in the delivery of successful systems and software projects, discipline is also required by standards, regulations, and quality improvement initiatives. Creating and managing requirements is a challenge of IT, systems and product development projects or indeed for any activity where you have to manage a contractual relationship. “ONLINE COURSE MANAGEMENT SYSTEM” undertaken has a project. This project is a web-based application specially for aspiring learners and instructors. This system mainly deals to manage the online courses for users. This application has been designed and developed a comprehensive web-based system to better support the students benefit and requirement. By using this application will make the students less wasted time, and they will be able to easily get all the studies information from their teachers. As a result, they will be able to continue their studies better. This take a look at develops an adaptive e-mastering platform wherein the learner is permitted to reply questions or resolve issues primarily based totally on potential or pace. The software program will undertake the Agile model. To accomplish this project, MERN Stack is used in the project. This project helps in safety, quality, and overall standard of learning amongst learners in educational institutions.

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
1	AGILE MODEL	3
2	SYSTEM ARCHITECTURE	11
3	FLOW DIAGRAM	16
4	USECASE DIAGRAM	17
5	SEQUENCE DIAGRAM	18
6	DEPLOYMENT DIAGRAM	19
7	ACTIVITY DIAGRAM	20
8	REGISTRATION PAGE	21
9	LOGIN PAGE	21
10	RESET PASSWORD	22
11	HOME PAGE	22
12	SEARCH INTERFACE	23
13	COURSE PAGE	24
14	BOOKMARK PAGE	25
15	PREFERENCE PAGE	25
16	INSTRUCTOR PAGE	26
17	COURSE ENTRY PAGE	27
18	VIDEO ENTRY PAGE	28
19	TERMINAL I	62
20	TERMINAL II	63

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	i
	LIST OF FIGURES	ii
1	INTRODUCTION	1
1.1	AGILE MODEL	2
2	SYSTEM SPECIFICATION	4
3	SOFTWARE DESCRIPTION	5
3.1	MERN STACK	5
3.2	MONGODB	6
3.3	EXPRESS	7
3.4	REACT	8
3.5	NODE	9
4	SYSTEM ANALYSIS	10
4.1	EXISTING SYSTEM	10
4.2	PROPOSED SYSTEM	10
4.3	SYSTEM ARCHITECTURE	10
4.4	SYSTEM OBJECTIVES	11
5	SYSTEM MODULES	13
5.1	LEARNER MODULE	13
5.2	INSTRUCTOR MODULE	13

5.3	WORKING EXPLANATION	14
5.4	USECASE DIAGRAM	17
5.5	SEQUENCE DIAGRAM	18
5.6	DEPLOYMENT DIAGRAM	19
5.7	ACTIVITY DIAGRAM	20
6	RESULTS AND DISCUSSION	21
6.1	OUTPUT SNAPSHOTS	21
6.2	SNAPSHOT DISCUSSION	28
7	CONCLUSION AND FUTURE ENHANCEMENT	31
7.1	CONCLUSION	31
7.2	FUTURE ENHANCEMENT	31
	APPENDICES	32
	APPENDIX I	32
	APPENDIX II	62
	REFERENCES	64

CHAPTER 1

INTRODUCTION

The technology of college students born in the virtual era brings an extensively exclusive studying technique. The mixture of technology and brand new technology's digitally-improved cognitive and social competencies will want new answers to the modern-day idea of learning. These are performed with the aid of using this "Online Course Management System" and, additionally, it offers a smooth consumer interface and security. Online Course Management System has been created with the goal of offering scholars and colleges a less difficult and handy way to impart knowledge. The features of this gadget are completed partially manually. Automated features to play a position in the shape of the gadget. Web-primarily based totally direction control gadget makes E-Learning possible. Ease of use and agency structure are its foremost characteristics. What makes the gadget precise is the hard and fast incorporated answers provided. It optimizes and strengthens the software of each module to the max, consisting of a coaching control platform. The open environment for growing web-primarily based totally training gadget, in addition to its adaptability and collaboration with the enterprise trendy, makes numerous groups successful in increasing or customizing their purposeful modules in step with requirements, which brings an interconnected and interactive web-primarily based totally studying surroundings to reality. Focusing on publications, web-primarily based totally direction control gadget integrates coaching and studying surroundings. Teachers can put online publications on the platform and college

students can pick publications and take a look at with the aid of using themselves. A usual Course Management System is a community gadget that's capable of organizing, presenting, managing, examining the contents of the publications and the coaching activities, and selling the interplay among college students and teachers. It is a specialty of assisting schools and universities construct an interactive website- primarily based on studying surroundings wherein anybody is capable of browsing the contents, acquiring resources, examining training impact, and cooperating with every different at any time.

1.1 AGILE MODEL

To overcome the limitation of the waterfall model, the agile model was introduced. Agile model is the combination of iterative and incremental software development model. In the agile model, the requirements are break up into many parts, called iterations, and then developed incrementally. In this model, each iteration is planned, designed, implemented, tested and deployed to the customers to take the feedback. If any changes required then the modification is done at that iteration then carry on the project. Any error can be fixed at each iteration so there is no issue about presence of errors in the project.



FIGURE 1 AGILE MODEL

- To establish close contact with the customer during development and to gain a clear understanding of various requirements, each Agile project usually includes a customer representative on the team. At the end of each iteration stakeholders and the customer representative review, the progress made and re-evaluate the requirements.
- Frequent delivery of incremental versions of the software to the customer representative in intervals of few weeks.
- Requirement change requests from the customer are encouraged and efficiently incorporated.
- It emphasizes on having efficient team members and enhancing communications among them is given more importance.

CHAPTER 2

SYSTEM SPECIFICATION

HARDWARE REQUIREMENT:

- Processor: 25MHz x86
- Hard Disk: Processor 250 GB or Higher
- RAM: 4 GB (Min)
- Storage Space: 2GB (Min)
- Laptop, Desktop &all types of Smart Phone.

CHAPTER 3

SOFTWARE DESCRIPTION

SOFTWARE REQUIREMENT:

- Operating System: Windows 7 or Higher
- Languages used: MERN Stack
- Tools: Visual Studio Code

3.1 MERN

MERN Stack is a collection of powerful technologies and robust, used to develop scalable master web applications comprising **backend**, **front-end**, and **database components**. It is JavaScript that is used for the faster and easier development of full-stack web applications. MERN Stack is a technology that is a user-friendly full-stack JavaScript framework for building applications and dynamic websites. MERN Stack consists of four main components or can say four main technologies:

1. **M** stands for **MongoDB (Database)**, mainly used for preparing document database and is a NoSQL (Non-Structured Query Language) Database System
2. **E** stands for **Express**, mainly used for developing Node.js web framework
3. **R** stands for **React**, mainly used for developing a client-side JavaScript framework
4. **N** stands for **Nodejs**, mainly used for developing the premier JavaScript web server

Each of these four technologies plays an important role in providing an end-to-end framework for the developers. Even these four technologies play an important role in the development process of web applications.

3.2 MongoDB

MongoDB is a document database designed for ease of application development and scaling. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time. The document model maps to the objects in your application code, making data easy to work with. Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyse your data. MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use. MongoDB is free to use.

ADVANTAGES:

- High availability through built-in replication and failover
- Horizontal scalability with native sharding
- End-to-end security
- Native document validation and schema exploration with Compass
- Management tooling for automation, monitoring, and backup
- Fully elastic database as a service with built-in best practices

3.3 Express

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application. It's a layer built on the top of the Nodejs that helps manage servers and routes. Express was created to make APIs and web applications with ease. It saves a lot of coding time almost by half and still makes web and mobile applications are efficient. Another reason for using express is that it is written in JavaScript as JavaScript is an easy language even if you don't have a previous knowledge of any language. Express lets so many new developers enter the field of web development. The reason behind creating an express framework for Nodejs are: Time-efficient, Fast, Economical, Easy to learn, Asynchronous.

ADVANTAGES:

- Express is Unopinionated, and we can customize it.
- For request handling, we can use Middleware.
- A single language is used for frontend and backend development.
- Express is fast to link it with databases like MySQL, MongoDB
- Express allows dynamic rendering of HTML Pages based on passing arguments to templates.

3.4 REACT

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”. React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug. Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM. React can also render on the server using Node and power mobile apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

ADVANTAGES

- Uses virtual DOM which is a JavaScript object. This will improve apps performance, since JavaScript virtual DOM is faster than the regular DOM.
- Can be used on client and server side as well as with other frameworks.
- Component and data patterns improve readability, which helps to maintain larger apps.

3.5 NODE

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

ADVANTAGES

- Efficient performance.
- Easier development process.
- Reusable code.
- Ability to handle multiple requests.
- Ability to scale smoothly.
- Prompt code execution.
- Asynchronous and event-driven

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM:

This section discusses some of the traditional system that is offline education has been a part of our education system. In the existing system, the learner has to proceed the self-learning with books and very less instructor for the domain will be available. The learner finds difficulty in clarifying the doubts and at the same. There is no surety to continue the learning of their domain with limited number of books and less advancements.

4.2 PROPOSED SYSTEM:

The proposed system will make the learner to learn more and at the same time, the instructor will be available to clarify the doubts and to resolve the issues of the learner. Instructor can share the knowledge adversely to the world. Learner can continue the learning and master the domain with more skills.

4.3 SYSTEM ARCHITECTURE

Online Course Management System creates an integral teaching environment. Learners can browse the courses on the site, register the course, and so on, which represents a complete teaching process. Instructors are able to manage their teaching work, prepare courseware, arrange content, answer learner's questions. The platform is a powerful learning tool which

supports customization by learner's need. And students can choose the courses they want to take, arrange their learning plans. The platform is also a bridge of communication between learner and instructor.

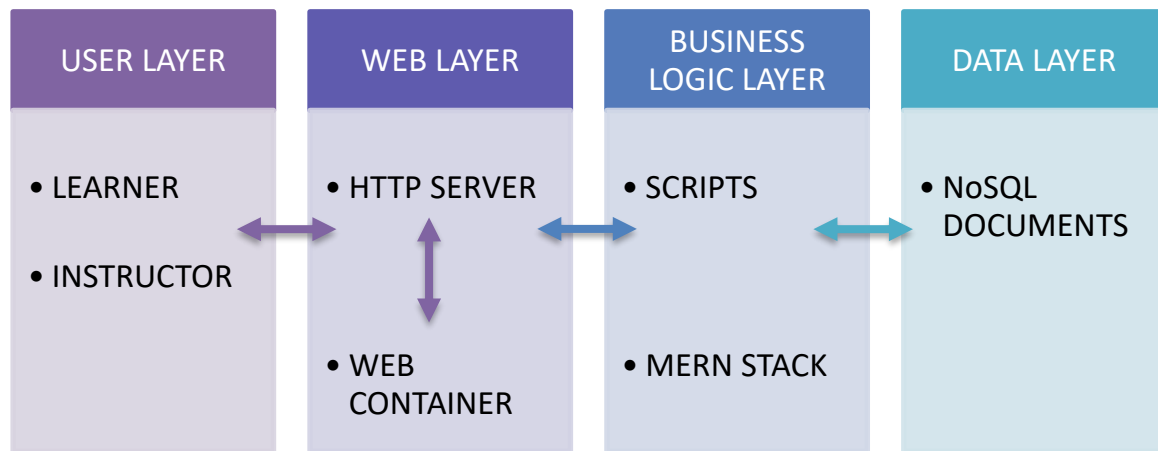


FIGURE 2 SYSTEM ARCHITECTURE

4.4 SYSTEM OBJECTIVES

- Every user needs to complete the registration with verified mail ID for security purposes.
- Easy way of accessing the website with login portal.
- This site makes the user to feel the easy user-interface mode with all delight features.
- The user can enroll in the course and start the learning in their own pace.
- The courses are placed in their respective categories, so search is quick.

- Courses are bookmarkable for easy accessing.
- The users can register themselves as instructors to share their knowledge.
- Instructors can make the content and the students to learn from them.
- Comment section is available where doubts and issues can be cleared.
- Rating the enrolled courses is available for students.
- Instructors can edit and delete the courses.
- Develop a feature which will be used to make sure that all the levels of learning are covered in a course.
- Recommended Courses are used here to showcase the preferred courses of the learner.

CHAPTER 5

SYSTEM MODULES

DESCRIPTION OF MODULES

- Learner Module
- Instructor Module

5.1 LEARNER MODULE

This module is for the aspiring learners where learners can register themselves and start choosing the course that needed. Learner can enroll into the course and start the learning at their own pace. Learner can set their preferences and the recommended courses paves the way for their domain. Learner can comment out the doubts and issues that to be clarified and resolved.

5.2 INSTRUCTOR MODULE

This module is for the instructors where the instructor can upload the course as they are there to share the knowledge to the aspiring learners. Instructor can also edit and delete the course. Instructor can also learn from another courses and there is no separate portal. Thus, Instructor can learn and also share the knowledge to the learners.

5.3 WORKING EXPLANATION

The user has to register themselves with appropriate Name, Email ID and, Password which satisfies the constraints. If any issue in matching the constraints, the warning will be thrown immediately. The registration will be completed only if Email ID is verified through OTP. If registration is done without OTP verification, then the login will fail. To gain the access for login, the user has to verify the Email ID by the OTP verification for the same Email ID. And then, user can login into the website with their Email ID and password. If login fails, appropriate warning will be thrown. The warning helps the user to move further. The sessions are passed for each login. The session time for each login is 5 hours. The user can go through the courses available in the website. The user can search the course by the course name or by the instructor's name. The user can also check the courses by the categories. If user want to learn from the course, the user can enroll in the course and start their learning pace. The user can bookmark the courses for ease access of the courses. Each course has progress bar, which denotes the completion of the course. The user can enable and set out the preferences and check all the recommended content in the homepage. The user can rate the course by providing the stars. And also, the user can comment their issues and doubts. So, instructor can clarify the doubts and also try to resolve the issues. The user can share the knowledge by enrolling as instructor and the user can create the course with refreshing content. The instructor can add, edit and delete the course. The instructor can make videos and share them in the courses. Effectivity of progress bar is shown only if videos are seen by the learner. The instructor can publish the courses with pre-requisites

and some words to inspire the learners. The course introduction is provided for all courses by the instructor. The learner can start the course after referring all the introduction of the course. The user and instructor can logout from the site after the using the site. If not, the session maintenance will handle the logging off from the session when the session time meets it. As a result, the goal was to simplify the students study easier in a secured manner with easy user-interface and move them forward.



FIGURE 3 FLOW DIAGRAM

5.4 USECASE DIAGRAM

A use case model describes different types of users interacts and their activities into a system. It is also a list of action which done by the user. The number of elements like an actor, an event, a usecase. The major component is the actor.

In this system, there are 2 actors.

- Learner
- Instructor

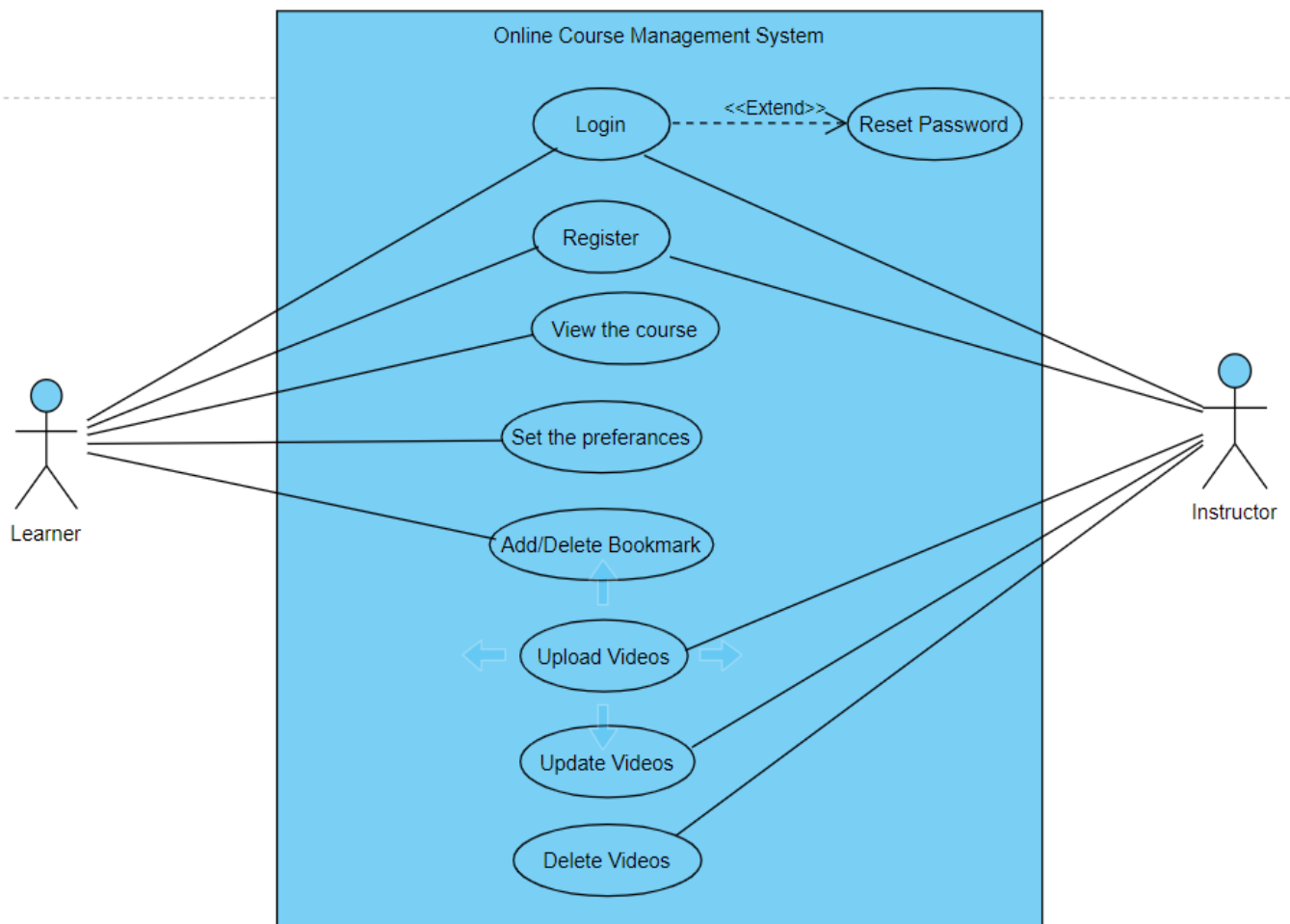


FIGURE 4 USECASE DIAGRAM

5.5 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

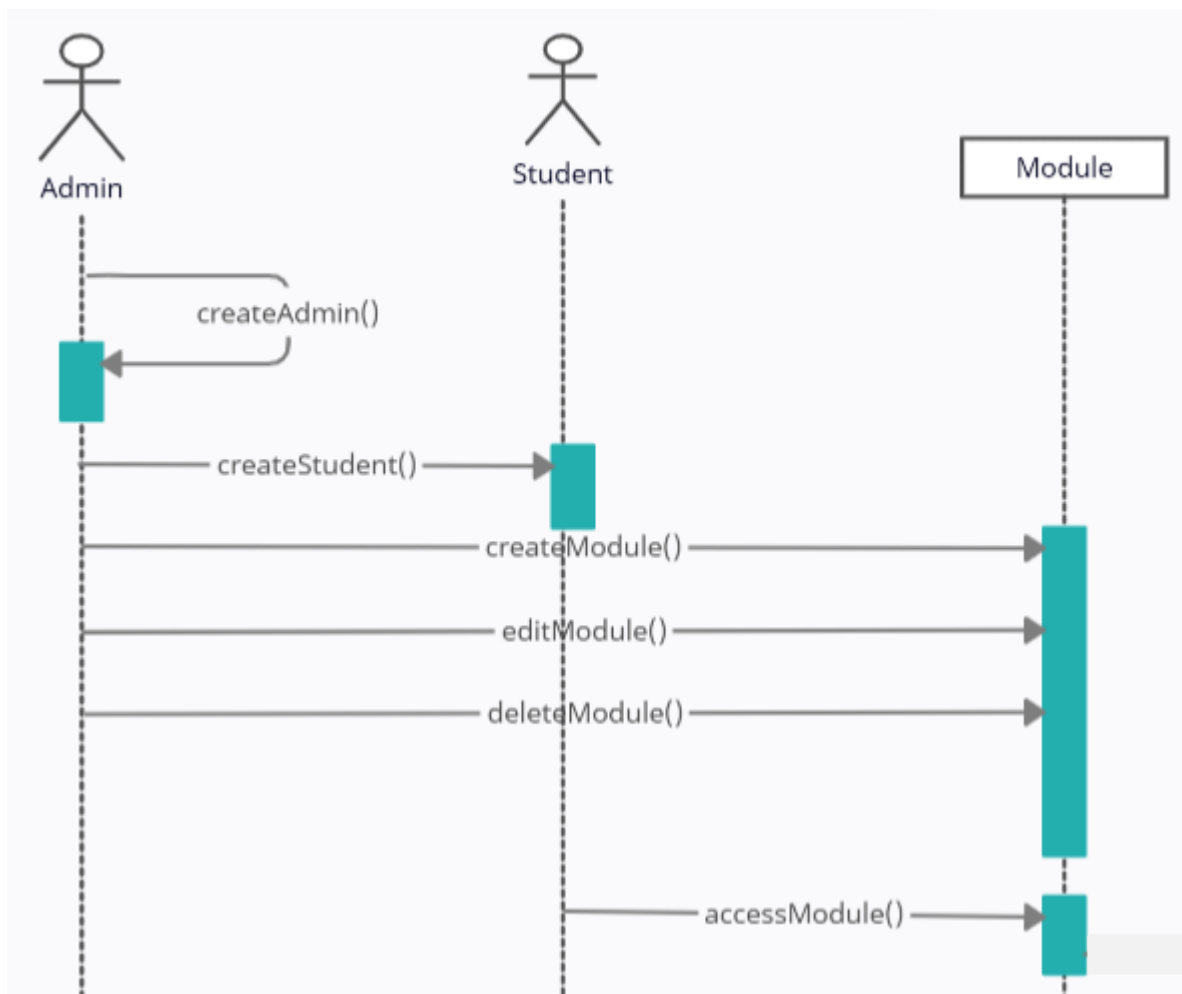


FIGURE 5 SEQUENCE DIAGRAM

5.6 DEPLOYMENT DIAGRAM

A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system. Using it you can understand how the system will be physically deployed on the hardware.

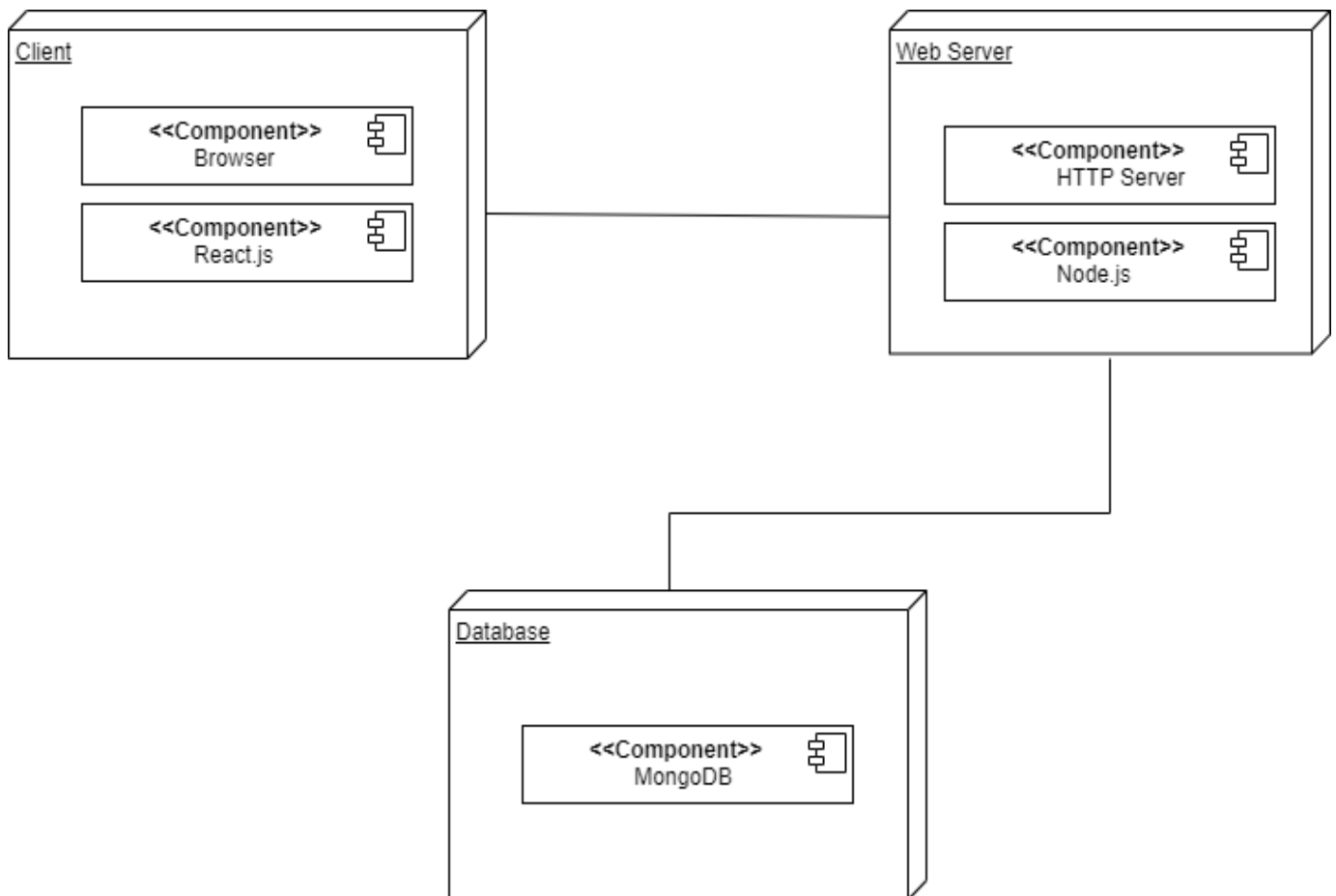


FIGURE 6 DEPLOYMENT DIAGRAM

5.7 ACTIVITY DIAGRAM

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

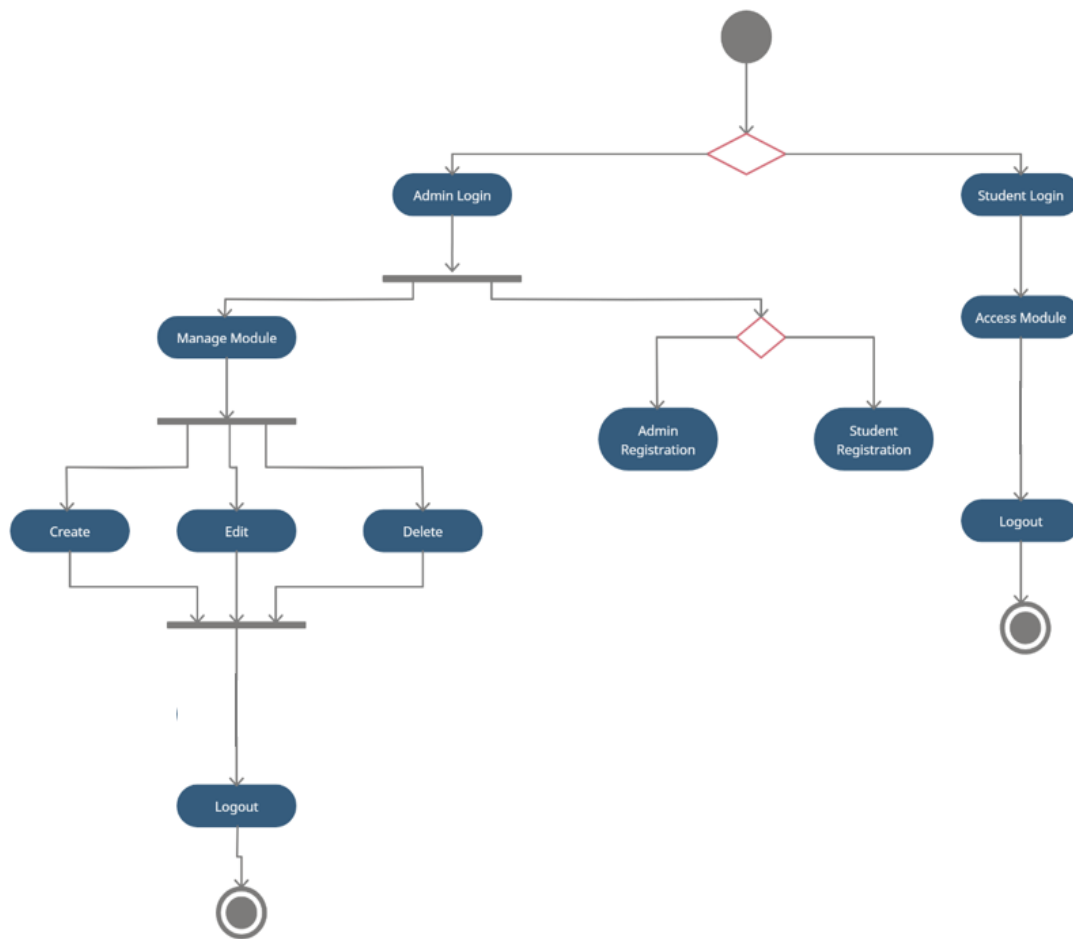
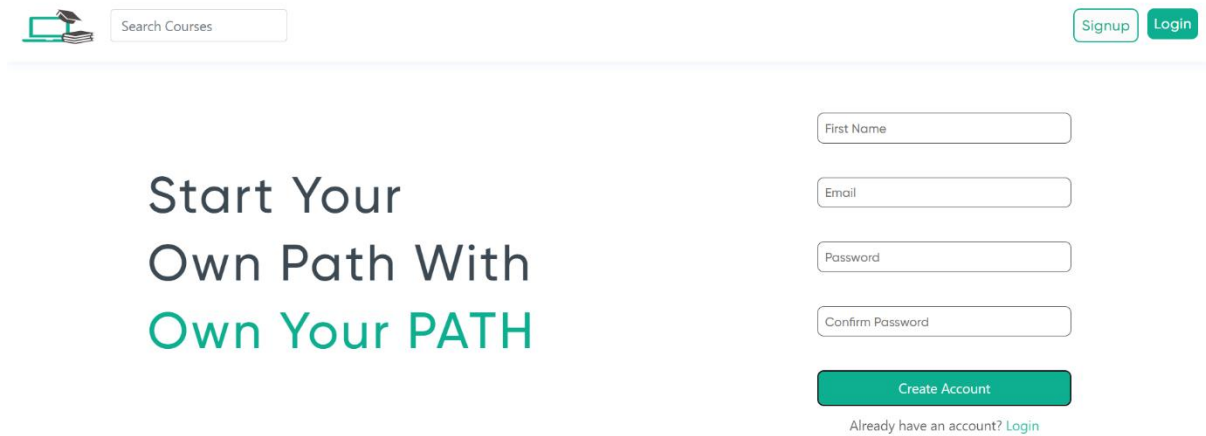


FIGURE 7 ACTIVITY DIGRAM

CHAPTER 6

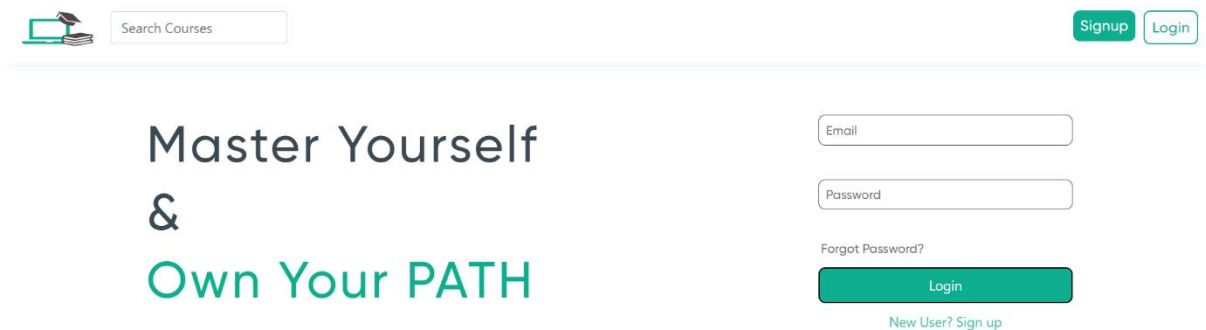
RESULT AND DISCUSSION

6.1 OUTPUT SNAPSHOTS



This is a screenshot of a registration page. At the top left, there is a search bar with a laptop icon and the text "Search Courses". At the top right, there are two buttons: "Signup" and "Login". The main heading on the left reads "Start Your Own Path With Own Your PATH". On the right, there are four input fields: "First Name", "Email", "Password", and "Confirm Password". Below these fields is a green "Create Account" button. At the bottom right, there is a link that says "Already have an account? Login".

FIGURE 8 REGISTRATION PAGE



This is a screenshot of a login page. At the top left, there is a search bar with a laptop icon and the text "Search Courses". At the top right, there are two buttons: "Signup" and "Login". The main heading on the left reads "Master Yourself & Own Your PATH". On the right, there are two input fields: "Email" and "Password". Below these fields is a link that says "Forgot Password?". At the bottom right, there is a green "Login" button. At the very bottom right, there is a link that says "New User? Sign up".

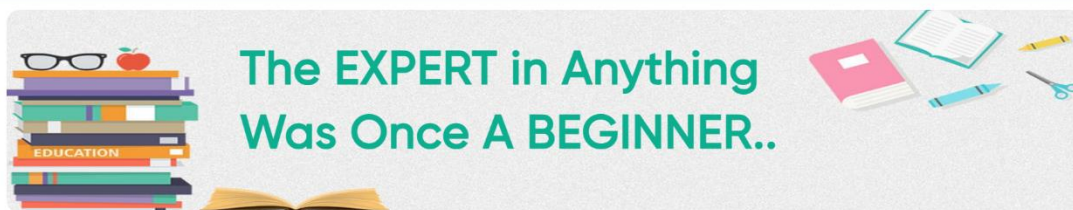
FIGURE 9 LOGIN PAGE

[Signup](#)[Login](#)

Reset Password, Enter your Email

[Back to Login](#)[Submit](#)[New User? Sign up](#)

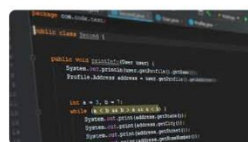
FIGURE 10 RESET PASSWORD PAGE

[Start Tutoring](#)[Bookmark](#)[Logout](#)

Categories

[All Courses](#)[Web Development](#)[Programming Languages](#)[AI / ML](#)[Cloud Development](#)[Data Science](#)[Recommended!](#)

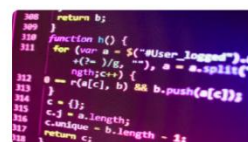
Welcome Monica S!



Java

Monica S

3 ★★★★★ (4 ratings)



C Programming

Monica S

3 ★★★★★ (5 ratings)

Get Course Recommendation
according to your Interest

[Choose Interest](#)

FIGURE 11 HOMEPAGE

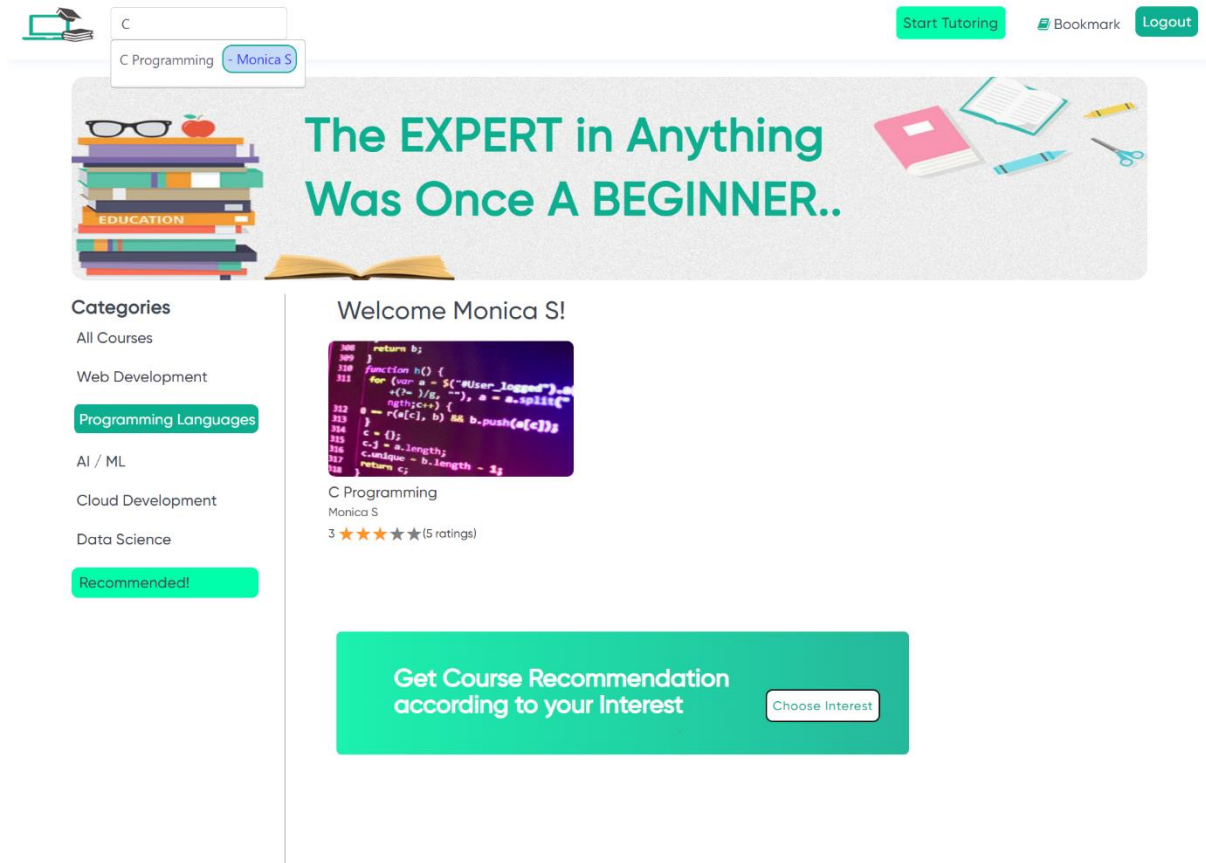


FIGURE 12 SEARCH INTERFACE

[Start Tutoring](#)[Bookmark](#)[Logout](#)

Java

3 ★★★★★ (4 ratings)

Complete Java Course!

Created at 2022-02-11

By Monica S



About

Requirement of this Course

Nothing

Descripton

Learn More as you can. Be a strong java programmer.
Be a beginner and Expertise finally.

What will you learn from this course?

Java Basics

OOPS

Collections

Data Structures

Video 0

4.22

Video 1

4.22

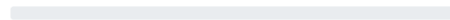
Video 2

4.22

Video 3

4.22

You have Completed 0.0% of your course!



Rate the course here please



FIGURE 13 COURSE PAGE

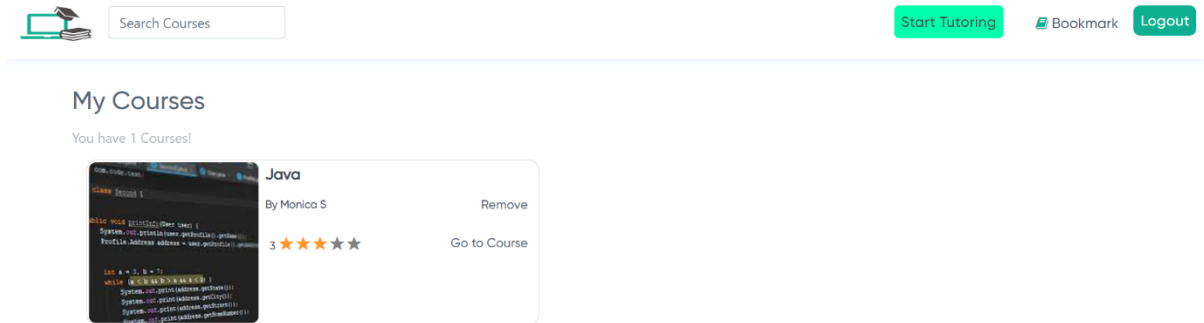


FIGURE 14 BOOKMARK PAGE

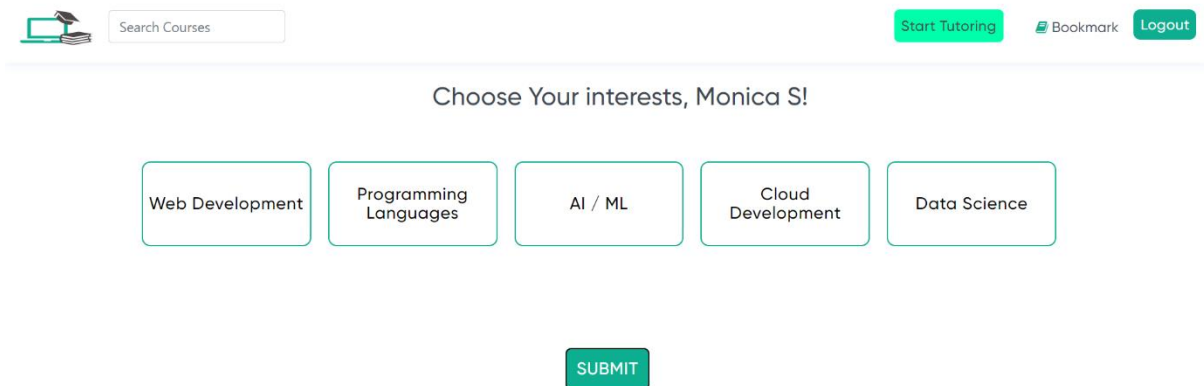


FIGURE 15 PREFERENCE PAGE

[Start Tutoring](#)[Bookmark](#)[Logout](#)

Are You EXPERT in Anything?
Here is Your PATH..

[Create New Course](#)

Dashboard

[Upload Your Courses](#)[Edit your Course](#)

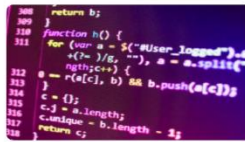
Here are your courses, Monica S!



Java

Monica S

3 ★★★★★



C Programming

Monica S

3 ★★★★★

FIGURE 16 INSTRUCTOR PAGE



Search Courses

Start Tutoring

Bookmark

Logout

Welcome Monica S!

Enter your Name

Your Name

Title

Enter Course Title

Course Category

Web Development

Programming Languages

AI / ML

Cloud Development

Data Science

Description of your Course

Short Description

eg: Complete HTML5, CSS3, Basics of Js

Long Description

Paragraph **B** *I*

What will the students learn from this?

Paragraph **B** *I*

What are the requirements of this course?

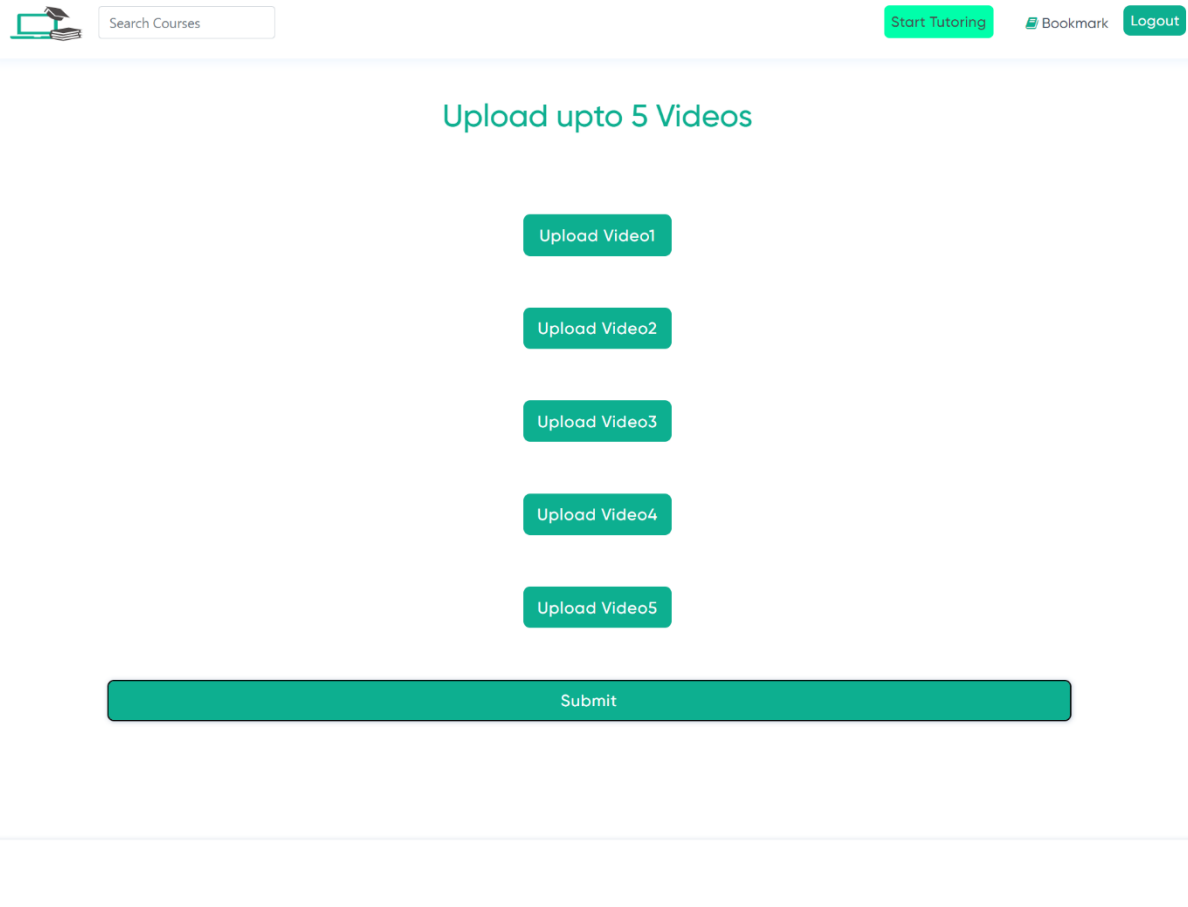
Paragraph **B** *I*

Upload Image

No file
Selected

Next

FIGURE 17 COURSE ENTRY PAGE



The screenshot shows a web interface for video uploads. At the top, there is a navigation bar with a search icon and a 'Search Courses' input field on the left, and 'Start Tutoring', 'Bookmark', and 'Logout' buttons on the right. The main heading is 'Upload upto 5 Videos'. Below this, there are five green buttons labeled 'Upload Video1', 'Upload Video2', 'Upload Video3', 'Upload Video4', and 'Upload Video5' arranged vertically. At the bottom of this section is a wide green button labeled 'Submit'. The page is separated by horizontal lines.

FIGURE 18 VIDEO ENTRY PAGE

6.2 SNAPSHOT DISCUSSION

- **Registration Page**

Registration page for learners and instructors with fields of Name, Email ID, Password. And OTP will be sent to Email ID for verification.

- **Login Page**

Login page is for the all verified users with fields of Email ID and Password. If the users are not verified, then user have to register to start their learning pace.

- **Reset Password Page**

Reset Password page is used if user forgets the password. But user must have registered to get change their password.

- **Home Page**

Home Page is the main dashboard for the users. Here, all course cards are displayed for the users to encourage their learning.

- **Search Interface**

Here, Search Bar is placed in the navigation bar for easy retrieval of courses by the course name or by instructor name. And also, categorical search is available.

- **Course Page**

Here, the page contains relevant details of the course. Course description, Pre-requisites, and Outcome of the course will be present in the page.

- **Bookmark Page**

Here, bookmarked courses will be listed for ease access of the courses.

- **Preference Page**

The user can opt for the categories where all those categorical courses will be under “Recommended Courses”

- **Instructor Page**

Instructor Page is a page where all courses of the instructor will be displayed. The instructor can edit and delete the course from the listed courses if necessary.

- **Course Entry Page**

In this page, all necessary details of the course will be given for creating the course content.

- **Video Entry page**

Here, course videos are uploaded for the course.

CHAPTER 7

CONCLUSION AND FUTURE ENHACEMENT

7.1 CONCLUSION

Although the ambition was to achieve a complete system that will have a highly accuracy. This project was managed to develop a system and a guideline to on how an application can be developed for course management system based on web. The project had errors and fixed it. This application interface is simply and well organized so that the user can easily cooperate with it. Thus, learner and instructor are benefitted from this project and at the same time.

7.2 FUTURE ENHANCEMENT

The system has been developed with future development possibilities in consideration. This project also has the future scope of enhancement such as:

- More attractive user interface to make it more user friendly.
- Availability of payment option.
- Availability to add, edit and, delete the assessments for the course.
- To implement an artificial intelligence.

APPENDICES

APPENDIX 1

SOURCE CODE

BACKEND OF THE PROJECT:

app.js

```
const path = require('path');

const http = require('http');

require('dotenv').config()

const redis = require('redis');

const mongoose = require('mongoose');

const express = require('express');

const bodyParser = require('body-parser');

const api_key =require('./config/config');

const authRoutes = require('./routes/auth');

const teacherRoutes=require('./routes/teacher')

const homeRoutes= require('./routes/homepage')

const courseRoutes=require('./routes/coursepage')

// const {addRoom,getUser} = require('./chat');

const MONGODB_URI =api_key.mongo;

const app = express();
```

```

const server = http.createServer(app);

app.use(bodyParser.json());

app.use(express.static(path.join(__dirname, 'public')));

app.use('/images',express.static(path.join(__dirname, 'images')));

app.use('/videos',express.static(path.join(__dirname, 'videos')));

app.use((req, res, next) =>{ // To remove CROS (cross-resource-origin-platform) problem

  res.setHeader('Access-Control-Allow-Origin',"*"); // to allow all client we use *

  res.setHeader('Access-Control-Allow-

Methods',"OPTIONS,GET,POST,PUT,PATCH,DELETE"); //these are the allowed

methods

  res.setHeader('Access-Control-Allow-Headers', "*"); // allowed headers (Auth for extra

data related to authoriaztiom)

  next();

})

app.use(authRoutes);

app.use(teacherRoutes);

app.use(homeRoutes);

app.use(courseRoutes);

if (process.env.NODE_ENV !== 'test') {

  mongoose

    .connect(MONGODB_URI,{ useUnifiedTopology: true,useNewUrlParser: true })

```



```
.then(()=> {  
    server.listen(8080);  
    console.log("Server Started!")  
})  
.catch(err => {  
    console.log(err);  
});  
}  
module.exports = app;
```

/routes/auth.js

```
const express = require('express');  
const authController =require('../controllers/auth');  
const { check } = require('express-validator');  
const Auth = require('../Authentication/is-auth');  
const router= express.Router();  
const User=require('../model/user');  
router.post('/signup',[  
    check('email')  
    .isEmail()  
    .withMessage('Please enter a valid email')
```

```

.custom((value,{req})=>{
    return User.findOne({email:value})
    .then(user=>{
        if(user){
            return Promise.reject('Email already exists!');
        }
    })
}),
check('password')
    .trim()
    .isLength({min:5}),
check('name')
    .trim()
    .not()
    .isEmpty()
],authController.signup);
router.post('/login',[
    check('email')
    .isEmail()
    .withMessage('Please enter a valid email')
    .custom((value,{req})=>{

```

```

    return User.findOne({email:value})

    .then(user=>{

        if(!user){

            return Promise.reject('No account with this email !');

        }

    })

    ]],authController.login);

router.post('/signup/otp',authController.otpVerification);

router.post('/signup/resetOtp',authController.resetPassword);

router.post('/signup/otp-resend',authController.resendOtp)

router.post('/signup/checkOtp',authController.resetOtpVerification);

router.post('/signup/reset-password',authController.newPassword);

// Fetching access Token using refresh token

router.post("/auth/token/",Auth.GetnewAccessToken);

module.exports = router;

```

/routes/coursepage.js

```

const express = require('express');

const router= express.Router();

const courseController =require('../controllers/coursepage');

const Auth = require('../Authentication/is-auth');

```

```

router.get('/course/:courseName/:courseId',Auth.authentication,courseController.CoursePage);

router.post('/home/:courseId/:courseName',Auth.authentication,courseController.Bookmark);

router.get('/users/:userName/:userId',Auth.authentication,courseController.ShowBookmark);

router.post('/unbookmark',Auth.authentication,courseController.unbookmark);

router.put('/rating',Auth.authentication,courseController.rating);

module.exports = router;

```

/routes/homepage.js

```

const express = require('express');

const router= express.Router();

const homeController =require('../controllers/homepage');

const Auth = require('../Authentication/is-auth');

router.get('/home/allCourses',homeController.allCourses);

router.get('/home/:course',homeController.fetchCourses);

router.post('/home/interests/',Auth.authentication,homeController.getPreferences);

router.post('/home/:course',Auth.authentication,homeController.preferenceCourses);

module.exports = router;

```

/routes/teacher.js

```

const express = require('express');

const router= express.Router();

const multer = require('multer');

const teacherController =require('../controllers/teacher');

const Auth = require('../Authentication/is-auth');

const ImagefileStorage = multer.diskStorage({

  destination:(req,file,cb)=>{

    cb(null,'images');

  },

  filename: (req,file,cb)=>{

    cb(null, new Date().toDateString() + '-' + file.originalname)

  }

})

const ImagefileFilter=(req,file,cb)=>{

  if(file.mimetype ==="image/png" || file.mimetype=== "image/jpg" ||

file.mimetype=== "image/jpeg"){

    cb(null,true);

  }

  else {cb(null,false);

    console.log("wrong file type")}

}

```

```

const VideofileStorage = multer.diskStorage({
  destination:(req,file,cb)=>{
    cb(null,'videos');
  },
  filename: (req,file,cb)=>{
    const currentDate= new Date();
    cb(null, currentDate.toDateString() +'-'+ file.originalname)
  }
})

const VideofileFilter=(req,file,cb)=>{
  if(file.mimetype ==="video/mp4"){
    cb(null,true);
  }
  else {cb(null,false);
    console.log("wrong file type")}
}

const imageMulter =
multer({ storage:ImagefileStorage,fileFilter:ImagefileFilter}).single('image')

const videoMulter=multer({ storage:VideofileStorage,fileFilter:VideofileFilter}).any()

router.post('/creator/create-course',imageMulter,teacherController.uploadCourse);

router.post('/creator/videoUpload/:courseID',videoMulter,teacherController.uploadVideo);

```

```
router.post('/creator/homepage',Auth.authentication,teacherController.teacherHome);  
router.post('/course/delete',Auth.authentication,teacherController.deleteCourse);  
router.post('/course/edit',Auth.authentication,teacherController.editCourse);  
router.put('/course/Update',imageMulter,teacherController.updateCourse)  
router.post('/watchedByuser',teacherController.watchedByUsers)  
module.exports = router;
```

FRONTEND OF THE PROJECT:

App.js

```
import React, {Component} from 'react';  
import {Route, Switch, Redirect} from 'react-router-dom';  
import {BrowserRouter} from 'react-router-dom';  
import Login from './Pages/Auth/Forms/Login/Login';  
import Signup from './Pages/Auth/Forms/Signup/Signup';  
import EmailVerify from './Pages/Auth/Forms/ForgotPassword/EmailVerify';  
import ForgotPasswordotp from './Pages/Auth/Forms/ForgotPassword/ForgotPassOtp';  
import ResetPassword from './Pages/Auth/Forms/ResetPassword/ResetPassword';  
import Cart from './Pages/Cart/Cart'  
import Otp from './Pages/Auth/Forms/Otp/Otp';  
import Homepage from './Pages/HomePage/Homepage';  
import TeacherPage from './Pages/Teacher/TeacherPage';
```

```

import TeacherVideos from './Pages/Teacher/TeacherVideos';

import TeacherHomePage from './Pages/Teacher/TeacherHomepage/TeacherHomepage';

import TeacherEdit from './Pages/Teacher/TeacherHomepage/TeacherEdit';

import CoursePage from './Pages/CoursePage/CoursePage';

import Preference from './Pages/HomePage/Preference';

class App extends Component {

  render(){

    return (

      <BrowserRouter >

        <Switch>

          <Route path="/signup" exact component={Signup}/>

          <Route path="/login" component={Login}/>

          <Route path="/signup/otp" component={Otp}/>

          <Route path="/forgotpasswordemail" component={EmailVerify}/>

          <Route path="/ForgotPasswordotp" component={ForgotPasswordotp}/>

          <Route path="/ResetPassword" component={ResetPassword}/>

          <Route path="/home/:CourseName" exact render={props =>

            <Homepage key={props.location.pathname} {...props}/>

          }/>

          <Route path="/home/Interest/Preference" exact component={Preference}/>

          <Route path="/course/:Course/:Courseid" exact render={props =>

            <CoursePage key={props.location.pathname} {...props}/>

          }/>

```



```

    <Route path="/Cart" component={Cart}/>

    <Route path="/Teacher" component={TeacherPage}/>

    <Route path="/TeacherVideos"
      render={(props)=> <TeacherVideos {...props}/> }/>

    <Route path="/TeacherHome" component={TeacherHomePage}/>

    <Route path="/TeacherEdit" component={TeacherEdit}/>

    <Redirect to="/home/all"/>

  </Switch>

</BrowserRouter>

);

}

}

export default App;

```

./ApiServices/auth.service.js

```

import axios from './axiosUrl';

class AuthServices {

  // ----- Authentication routes -----

  // fetching

  // RefreshToken(data){

  //   return axios.post('/auth/token',data);

```

```
// }

register(data) {

    return axios.post('/signup',data)

}

otp(data){

    return axios.post("/signup/otp",data)

}

otpResend(data){

    return axios.post('/signup/otp-resend',data)

}

login(data) {

    return axios.post('/login',data)

}

VerifyEmail(data){

    return axios.post('/signup/resetOtp',data);

}

VerifyOtp(data){

    return axios.post('/signup/checkOtp',data);

}

ResetPassword(data){

    return axios.post('/signup/reset-password',data);
```

```

}

logout(){
    localStorage.clear();
}

getCurrentUser(){
    return localStorage.getItem('user');
}

getUserName(){
    let userName=localStorage.getItem('userName');

    if(userName!=null)

        userName= userName.charAt(0).toUpperCase() + userName.slice(1);

    return userName;
}

// ----- end of auth routes -----

AllCourses(){
    return axios.get('/home/allCourses')
}

HomepageCourse(CourseLink){
    return axios.get(`/home/${CourseLink}`)
}

PreferenceCourse(CourseLink,data){

```

```

return axios.post(`/home/${CourseLink}`,data,{
  headers: {
    Authorization: 'Bearer '+ localStorage.getItem('user') + " " +
localStorage.getItem('ref_token')
  }
})
}

UpdatedCourse(data){
  return axios.put('course/Update',data);
}

//Bookmark

bookmarkCourses(userName,userId){
  return axios.get(`/users/${userName}/${userId}`,{
    headers: {
      Authorization: 'Bearer '+ localStorage.getItem('user') + " " +
localStorage.getItem('ref_token')
    }
  });
}

DeleteBookmark(data){
  return axios.post("/unbookmark",data,{

```

```

        headers: {

            Authorization: 'Bearer '+ localStorage.getItem('user') + " " +
localStorage.getItem('ref_token')

        }

    });

}

BookMark(CourseId,CourseName,data){

    return axios.post(`/home/${CourseId}/${CourseName}`,data,{

        headers: {

            Authorization: 'Bearer '+ localStorage.getItem('user') + " " +
localStorage.getItem('ref_token')

        }

    })

}

FetchCourses(CourseName,CourseId){

    return axios.get(`/course/${CourseName}/${CourseId}`, {

        headers: {

            Authorization: 'Bearer '+ localStorage.getItem('user') + " " +
localStorage.getItem('ref_token')

        }

    })
}

```

```
}
```

```
Rating(data){
```

```
return axios.put("/Rating",data,{
```

```
  headers: {
```

```
    Authorization: 'Bearer '+ localStorage.getItem('user')
```

```
  }
```

```
}})
```

```
TeacherHomePage(data){
```

```
return axios.post("/creator/homepage",data,{
```

```
  headers: {
```

```
    Authorization: 'Bearer '+ localStorage.getItem('user') + " " +
```

```
localStorage.getItem('ref_token')
```

```
  }
```

```
})
```

```
}
```

```
TeacherCourseDelete(data){
```

```
return axios.post("/course/delete",data,{
```

```
  headers: {
```

```
    Authorization: 'Bearer '+ localStorage.getItem('user')
```

```
  }
```

```
})
```

```
    }  
  }  
  
export default new AuthServices();
```

/ApiServices/axiosUrl.js

```
import axios from 'axios';  
  
import url from './BackendUrl';  
  
const instance = axios.create(  
  {  
    baseURL: url  
  }  
);  
  
instance.interceptors.response.use((response) => {  
  return response  
}, function (error) {  
  const originalRequest = error.config;  
  console.log(originalRequest);  
  if(error){  
    if (error.response.status === 401 && originalRequest.url === url + "auth/token") {  
      localStorage.clear()  
      window.location.href = '/login'
```

```

    return Promise.reject(error);
  }

  if (error.response.status === 401 && !originalRequest._retry) {

    originalRequest._retry = true;

    const refreshToken = localStorage.getItem('ref_token');

    if(refreshToken){

      console.log("calling reqqq")

      return axios.post(url + 'auth/token/', {refresh_token:refreshToken},

        {

          headers: {

            Authorization: 'Bearer ' + refreshToken

          }

        })

      .then(res => {

        if (res.status === 201 || res.status === 200) {

          console.log("sucess")

          localStorage.setItem("ref_token",res.data.refresh_token);

          localStorage.setItem("user",res.data.access_token);

          originalRequest.headers["Authorization"] = 'Bearer ' +

localStorage.getItem('user');

          return axios(originalRequest);

```



```

        }
    })
    .catch(err=>{
        console.log(err.response);
        localStorage.clear();
        window.location.href = '/login'
    })
}
else {
    console.log("No ref token was found")
    localStorage.clear();
    return window.location.href = '/login'
}
}
}

return Promise.reject(error);
});

export default instance;

```

/Components/UI/Input/FormInput.js

```
import React from 'react';
```

```

import './Input.css';

const input = (props)=> {

  let inputElement= null;

  var inputclasses =["InputElement"];

  if(props.invalid && props.touched){

    const index =inputclasses.indexOf('pop');

    if(index>-1) inputclasses.splice(index,1);

    inputclasses.push("Invalid");

  }

  else if(props.touched){

    const index =inputclasses.indexOf('pop');

    if(index>-1) inputclasses.splice(index,1);

    inputclasses.push("Valid"); }

  let error = <p>error</p>;

  if(props.msg!=="" && props.touched)

    error = <p className="text-success error-msg">{props.msg}</p>;

  if(props.errors!=="" && props.touched)

    error= <p style={{color: "red"}} className=" error-msg">{props.errors}</p>;

  else if(!props.touched)

    error=<p style={{opacity:"0"}}>a</p>;

  inputElement = <input

```

```

    type={props.type}

    placeholder={props.placeholder}

    className={inputclasses.join(' ')}

    value={props.value}

    onBlur={props.blur}

    onChange={props.changed}/>

    return(

        <div className="Input">

            {inputElement}

            {error}

        </div>

    );

}

export default input;

```

/Components/UI/Buttons/SubmitButton.js

```

import React from 'react';

const SubmitButton =(props)=> (

    <button className={props.className} type="Submit" >{props.Label}</button>

)

export default SubmitButton;

```

/Components/UI/Logo/Logo.js

```
import React from 'react';

import Logo from '../assets/Images/MonLogo.svg';

import './Logo.css';

const logo = () => (

  <img className="logo-oup" src={Logo} alt="logo"/>

)

export default logo ;
```

/Components/UI/Mainpage/MainPage.js

```
import React from 'react';

import './MainPage.css';

const mainPage = (props) => {

  let oup = null;

  if(props.oup){

    oup = (<>

      <br/><span className="heading-3">Own Your </span>

      <span className="heading-4">PATH</span>

    </>);

  }

}
```

```

    return(
      <div>
        <h1 className="Content-text"><span className="heading-
1">{props.heading1}</span>
        <br/><span className="heading-2">{props.heading2}</span>
        {oupt}
      </h1>
    </div>
  );
}
export default mainPage;

```

/Components/UI/Navigation/Navbar/Navbar.js

```

import React,{useEffect} from 'react';
import './Navbar.css';
import {NavLink,useHistory} from 'react-router-dom';
import Logo from '../UI/Logo/Logo';
import { GoogleLogout } from 'react-google-login';
import AuthServices from '../APIServices/auth.service';
import Search from '../Search/search';

```

```

const Navbar = ()=>{

  const [isLogin,setLogin]=React.useState(false);

  const history = useHistory()

  useEffect (()=>{

    let isMounted=true;

    if(isMounted){

      if(localStorage.getItem('user')){

        setLogin(true)

      }

    }

    return ()=>{

      isMounted=false;

    }

  },[])

  const logout=() => {

    setLogin(false)

    AuthServices.logout()

    console.log("logout called")

    history.push('/login')

  }

  let Logout = ( <ul className="navbar-nav ml-auto">

```

```

    <li className="nav-item">
      <NavLink to="/signup" activeClassName="btnactive" className="nav-link
Signupbtn">Signup</NavLink>
    </li>
    <li className="nav-item">
      <NavLink to="/login" activeClassName="btnactive" className="nav-link
Loginbtn">Login</NavLink>
    </li>
  </ul>
);
let loggedIn = (<ul className="navbar-nav ml-auto">
  <li className="nav-item" data-toggle="tooltip" data-placement="top"
  title="Create Your Course">
    <NavLink to="/teacherhome" activeClassName="teacherActive"
    className="nav-link teachLink">Start Tutoring</NavLink>
  </li>
  <li className="nav-item">
    <NavLink to="/Cart" className="nav-link ">
      <i data-toggle="tooltip" data-placement="top" title="Bookmarked Courses"
      className="fa fa-book" aria-hidden="true"><span id="bookmarkNav">
Bookmark</span></i></NavLink>

```

```

</li>

<li className="nav-item">

  <GoogleLogout

    buttonText="Logout"

    render={renderProps => (

      <NavLink to="/login" onClick={logout}

        disabled={renderProps.disabled} className="nav-link logoutlink" > Logout

</NavLink>

      )}

    onLogoutSuccess={logout}>

  </GoogleLogout>

</li>

</ul>

);

return(

  <nav className=" navbar navbar-expand-lg sticky-top ">

    <NavLink to="/home/all" className="navbar-brand"><Logo/></NavLink>

    <button className="navbar-toggler" type="button" data-toggle="collapse"

      data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"

      aria-expanded="false"

      aria-label="Toggle navigation">

```



```

      <i className="fa fa-bars" aria-hidden="true"></i>

    </button>

    <div className="collapse navbar-collapse" id="navbarSupportedContent">

      <Search/>

      { !isLogin && Logout }

      { isLogin && loggedIn }

    </div>

  </nav>

)

}

export default Navbar;

```

./Components/UI/Search/search.js

```

import React from 'react';

import { Link } from 'react-router-dom';

import styles from './search.module.css';

import { connect } from "react-redux";

function Search(props){

  const [query,setQuery]=React.useState(false);

  const [isOpen,setOpen]=React.useState(false);

  React.useEffect(() => {

```

```

let handler = (event) => {
  if (!searchNode.current.contains(event.target)) {
    setOpen(false);
  }
};

document.addEventListener("mousedown", handler);

return () => {
  document.removeEventListener("mousedown", handler);
};
}, []);

```

```

const queryHandler =(event)=>{
  setQuery(event.target.value);
  setOpen(true)
}

const filteredSubjects = (courses,query)=>{
  if(!query){
    return props.Courses;
  }
  else{
    return courses.filter(course=>{

```

```

        const title=course.title.toLowerCase();

        const name=course.name.toLowerCase();

        return title.includes(query.toLowerCase()) ||
name.includes(query.toLowerCase());

    })

}

}

let SearchItems = filteredSubjects(props.Courses,query);

// defining useRef hook

let searchNode = React.useRef();

return (

    <div>

        <form className="form-inline my-2 my-lg-0">

            <input className="form-control mr-sm-2" type="search"

                placeholder="Search Courses"

                aria-label="Search"

                onClick={()=>{ setOpen(true) }}

                onChange={(event)=>queryHandler(event)}/>

            { /* <button className="btn btn-outline-primary my-2 my-sm-0"

type="submit">Search</button> */ }

        </form>

```

```

    <div className={styles.searchItems} ref={searchNode}>

      {isOpen ? <ul>

        {SearchItems.map((item,index)=>{

          return <Link key={index} to={` /course/all/${item._id}`}

style={ { textDecoration:'none' } }><li className={styles.name} key={index}>

          {item.title}

          <span className={styles.author}>- {item.name}</span></li>

          </Link>

        }}}

      </ul>

      : null}

    </div>

  </div>

)

}

const mapStateToProps = (state) => {

  return {

    Courses: state.filter.Courses,

  };

};

export default connect(mapStateToProps, null)(Search);

```

APPENDIX II

```
PS E:\EE Project\OwnYourPath>
PS E:\EE Project\OwnYourPath> cd .\onlinecourse\
PS E:\EE Project\OwnYourPath\onlinecourse> cd .\Backend\
PS E:\EE Project\OwnYourPath\onlinecourse\Backend> npm start

> OwnYourPath@1.0.0 start E:\EE Project\OwnYourPath\onlinecourse\Backend
> nodemon app.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
(node:16420) DeprecationWarning: collection.ensureIndex is deprecated. Use createIndexes instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
Server Started!
█
```

FIGURE 19 TERMINAL I

```
PS E:\EE Project\OwnYourPath>
PS E:\EE Project\OwnYourPath> cd .\onlinecourse\
PS E:\EE Project\OwnYourPath\onlinecourse> cd .\Front-end\
PS E:\EE Project\OwnYourPath\onlinecourse\Front-end> npm start

> my-app@0.1.0 start E:\EE Project\OwnYourPath\onlinecourse\Front-end
> react-scripts start

Starting the development server...

./src/Pages/Auth/Forms/ResetPassword/ResetPassword.js
  Line 186:13:  'value' is assigned a value but never used  no-unused-vars
./src/Pages/Auth/Forms/Login/Login.js
  Line 23:36:  Unnecessary escape character: \.  no-useless-escape
./src/Pages/Auth/Forms/ForgotPassword/EmailVerify.js
  Line 24:36:  Unnecessary escape character: \.  no-useless-escape
  Line 157:9:  'value' is assigned a value but never used  no-unused-vars
./src/Pages/Auth/Forms/Signup/Signup.js
  Line 40:40:  Unnecessary escape character: \.  no-useless-escape

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

FIGURE 20 TERMINAL II

REFERENCES

1. <https://www.softwaretestinghelp.com/learning-management-system/>
2. <https://moodle.org/>
3. <https://www.iitms.co.in/blog/what-is-course-management-systems-for-higher-education.html>
4. <https://www.techtarget.com/searchcio/definition/learning-management-system>
5. <https://commercemates.com/project-report-meaning-characteristics-need-objectives/>
6. <https://cft.vanderbilt.edu/guides-sub-pages/course-management-systems/#:~:text=Resources-Introduction,as%20course%20syllabus%20and%20handouts>
7. <https://careerfoundry.com/en/blog/ux-design/the-difference-between-ux-and-ui-design-a-laymans-guide/>
8. <https://www.mongodb.com/mern-stack>
9. <https://www.mongodb.com/>
10. <https://expressjs.com/>
11. <https://nodejs.org/>
12. <https://reactjs.org/>
13. <https://www.npmjs.com/package/axios>