“Online Learning Market Place”

J-Component PROJECT REPORT

Submitted for the course: CSE3002 Internet and Web Programming

By

Rohit Krishna (18BCE0537)

Sai Srujith D (18BCE0560)

Slot: B2

Name of faculty: Dr. K. Jayakumar

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Novmember, 2020

CERTIFICATE

This is to certify that the project work entitled “Online Learning Market Place” that is being submitted by “Rohith Krishna (18BCE0537), Sai Srujith (18BCE0560)” for CSE3002 Internet and Web Programming is a record of bonafide work done under my supervision. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Place: Vellore

Date: 1/11/2020

Signature of Students:

Rohith Krishna

Rohith

Sai Srujith D

srujith

Signature of Faculty:

ACKNOWLEDGEMENTS

I take immense pleasure in thanking Dr. G. Viswanathan, my beloved Chancellor, VIT University and respected Dean, Dr. R. Saravanan, for having permitted me to carry out the project.

I express gratitude to my guide, Dr. K. Jayakumar, for guidance and suggestions that helped me to complete the project on time. Words are inadequate to express my gratitude to the faculty and staff members who encouraged and supported me during the project. Finally, I would like to thank my ever-loving parents for their blessings and my friends for their timely help and support.

Signature of Students:

Rohith Krishna

Sai Srujith D

INDEX PAGE

ABSTRACT

Learning has no age. And to help those curious learners from teenagers to the golden era, many e-learning platforms and apps like Udemy, Coursera, Lynda, MasterClass have come up with digital solutions.

What made these e-learning platforms successful is the wide range of courses and quality services they offer online at an affordable cost. Consider Udemy, an online learning and teaching app founded in 2010 has over 100,000 courses and 24 million students today. The demographic coverage and the demand for e-learning platforms are so high that the global growth of the mobile learning market is expected to be worth $37 billion in 2020.

In our project we created an online learning market place. Main aim of our project is to make learning more user-friendly. Anyone with no technical knowledge can use our site to make their learning easy. With simple user interface and user friendly features, our site is more easy to use for any kind of user. To also provide the user the verification that he has completed the course by providing him certificate of completion upon attending specified percentage of classes which is decided by the admin, and also we provide them with course content for each course they have chosen.

1. **INTRODUCTION**

1.1 **PROBLEM STATEMENT**

This project is aimed to develop a user friendly online learning market place , which Provide the user with options to register to the website and then access it. The user is given the options to buy courses offered by the organization along with ability to view the running courses presently and is also given the privilege to view the courses completed at any time user visits the page .User is provided with kart feature and certificate generation and certificate mailing options .Our projects solves the problem of many e-learning websites which do not have a clean user interface and also we introduce the concept of attendance , to get certificate of completion.

We achieve so by efficient use of **REST** API and perform create , read, update operations wherever required on the request of user

1.2 .**TECHNICAL SPECIFICATION:**

1.2.1 **FRONT END TECHNOLOGY:**

1.EJS

2.CSS

3.JQUERY

4.JAVASCRIPT

5.BOOTSTRAPS

1.2.2 **LATEST TECHNOLOGY:**

1.EJS.

2 Bootstraps

1.2.3 **BACKEND TECHNOLOGY:**

1.MONGO DB

1.2.4 **SERVER SIDE SCRIPTING :**

1.NODE JS

1.2.5 **SOFTWARE REQUIREMENTS:**

1. WINDOWS 7 OR HIGHER

2.VISUAL STUDIO CODE

2.**EXISTING SYSTEM PROBLEMS**

Online learning platforms provide a very good opportunity to students but they have a few problems which needs to be addressed.

First problem is availability. Most of the courses provided by these platforms are costly and well out of reach for many. In our project we have give an option for the user to choose between the free courses and paid courses. Anyone who doesn't want to pay, can directly view free courses. This reduces their burden for keep on searching for courses they can afford. Without browsing much, user can choose from the free courses. This makes our site much more user friendly.

Another problem with these applications is that, they trouble students with un-realistic deadlines. So often we come across students who stopped learning some course without finishing it. This is because of the deadlines that are put to finish the course. This discourages students, from learning and they tend to give up. In our project, we have give student full control over when to complete the course and how long he can take to finish it. There is no particular time period to finish it. This makes our project much more user friendly, which our main aim in this project

3.**PROPOSED SYSTEM DESIGN**

3.1 MODULES AND DESCRIPTION

3.1.1. **Register Module**

In this module , user who is new to the website creates his account by filling the form ,

The values entered in form are validated using javascript and also form verification was used to reduce possibilities of wrong entries.

The form then sends a post request to the route '/hmp' , where we use the **create** function to create a new object in database and it is stored

3.1.2.**Login module**

The returning user should remember the username and password , once they are entered, the form sends a post request to route '/login' , wherein we use the **findOne** function to find the object in the database which matches the credentials entered by the user , if match is found the page is redirected to userpage which uses id of the object stored in database as a parameter to retrieve the details of the user in object form as **mongodb** we use **nosql** , so the object is then passed onto the user page which contain details as mentioned in schema

3.1.2.**My Courses Module**

As mentioned in explanation of previous module , we retrieve the object from the database using id of the object as the parameter using function **findById** and then object is passed onto the userpage in form of **json** object , where we use them to display the essential details of the user as well as establish routes that shows the user the paid courses and courses in progress and courses completed by the user.

3.1.4. **Paid Courses Module**

User can view the paid courses that are available , so that he can register , by clicking on the buy button , then the course gets added in the kart of the user , on clicking the submit button the form sends a post request to route '/kartupd/:id/:cid' where cid is the course id created when course is added to **courses** collection and id is the id of the user created when user register(**regs collection**) before login, we use **findById** on **courses** collection , to obtain the object that matches in form of json object and then we create another json object with essential details and then we use function

findByIdAndUpdate on the **regs** collection(registered student record collection) and update an property called **kart** which is of type array that holds the json objects that contain details of the course selected by the user in buy page , then attributes required to show the course when the user decides to see the kart page

3.1.5. **My Cart Module**

As mentioned in previous module, the courses that buys are first moved to the kart of the user using the functions mentioned in the previous module that is **findByIdAndUpdate** , then the courses are seen in the kart , then once the user choses the buy option , the post request is sent to route

'/update/:id/:cn' ,where id is the parameter used to retrieve the object matching to it from the **regs** collection of the database and the cn is the course name used to retrieve the json object matching the course name using **findOne** function , we then create an new json object , which is then used to update the progCourses property which is a array of json objects each object representing the object of the **courses** collection which were selected by the user while buying the courses .

3.1.6. **In progress Module**

As mentioned in the previous module courses that are placed in kart , then when the user buys the course , the course gets added to progCourses property which is an array of json objects containing the users courses that are in progress , the update was made after the course and is of the user was duly extracted from the respective collections and **findByIdAndUpdate** function was used and progCourses property was alone updated using specific mongodb operation $push:{property:json object to be updated in the property} , then after this updation , we redirect to the page containing the progress courses page of the user , in this page can access the content of the courses and also the complete course option is enabled if the attendance of the user is more than 75% which is controlled by the admin.

3.1.7.**Admin module**

This module is responsible for attendance maintenance of the students who are registered on the portal , the admin can set the total number of classes , then update individual attendance of the users , and this parameter is used to calculate attendance which in turn enables the complete courses option, attendance updation is done using property of mongodb that allows us to update classes attended and total classes.

3.1.8. **Completed Courses Module**

This module is responsible for maintenance of completed courses,the courses are seen which are present in compCourses property of the **regs** collection , which is updated once the student attendance is more than 75% , then an post request is sent to route '/compcd/:id/:cn' where the id is id of the user assigned when record is created in **regs** collection at the beginning and the cn is the course name of the course in **courses** collections both are used to retrieve respective records in json form ,then update the property using mongo dB property

which enables us to update the compCourses property , which is reflected here , then user is provided with certificate generation

3.1.9. **Certificate generation and mailing module**

This module is responsible for certificate generation , here the user is given the option to download the certificate and mail it to email account in the details given by him during registration , which is retained from regs collection **nodemailer** library , we use **createTransport** function to configure the email from which you want to send the mail to the recipients and then we send it to the user using **sendMail** which helps in attaching attachments and mention the receiver's email retrieved from regs collection and hence send his certificate to registered email upon user's request.

3.1.10 .**Course Content Module**

This module is responsible for displaying the contents of the course present in the progress courses page of the user , when user chooses to , then we retrieve contents from the **coursecontent** collection and parameter used to retrieve content being course name by using function **findOne** and we retrieve contents on the webpage that is then redirected to

3.2 **Pseudo Code**

1.This program runs on a local host and on port 3000 as mentioned in the code.

2.Enter the url "http://localhost:3000/hmp" after server starts to redirect to home page

3. User can see a variety of options available some of them being login , register and admin login, choose an option

4.if option selected ==”register” :

user has to fill the form given to register with details asked for

validation is done using javascript, to prevent errenous entries

after filling details click on submit and user will be redirected to

login page

5. if option selected ==”login”:

Credentials are verified :

If matched->redirected to user page:

User can view the current ,paid and completed courses

Else-> back to login page

6.if option selected ==”admin”:

Credentials are verified:

If true -> redirected to admin portal

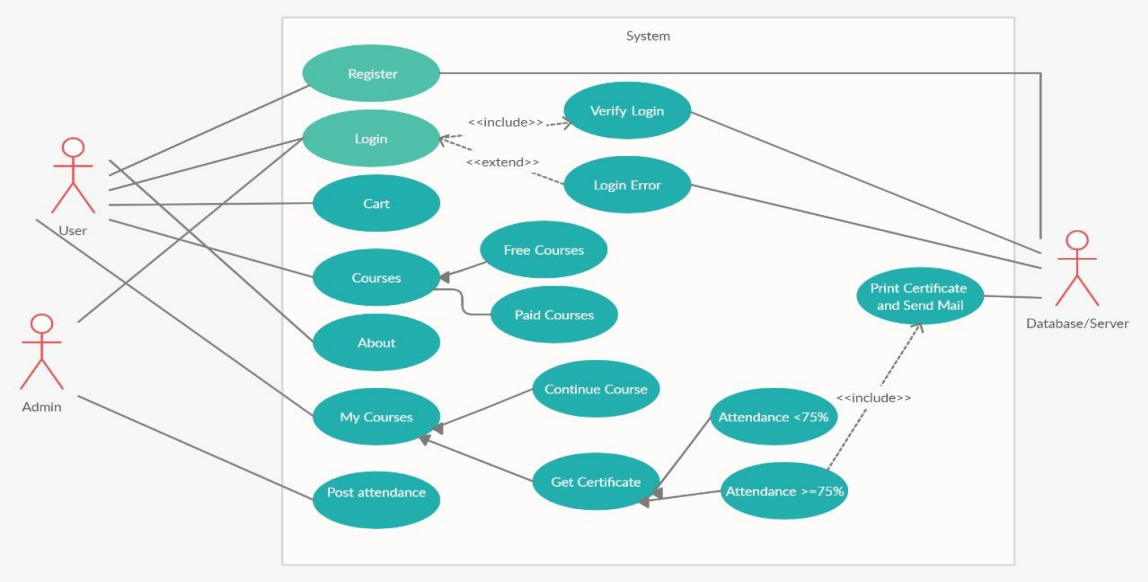
Attendance and registered user's are seen

Else ->redirected to admin login

7.End

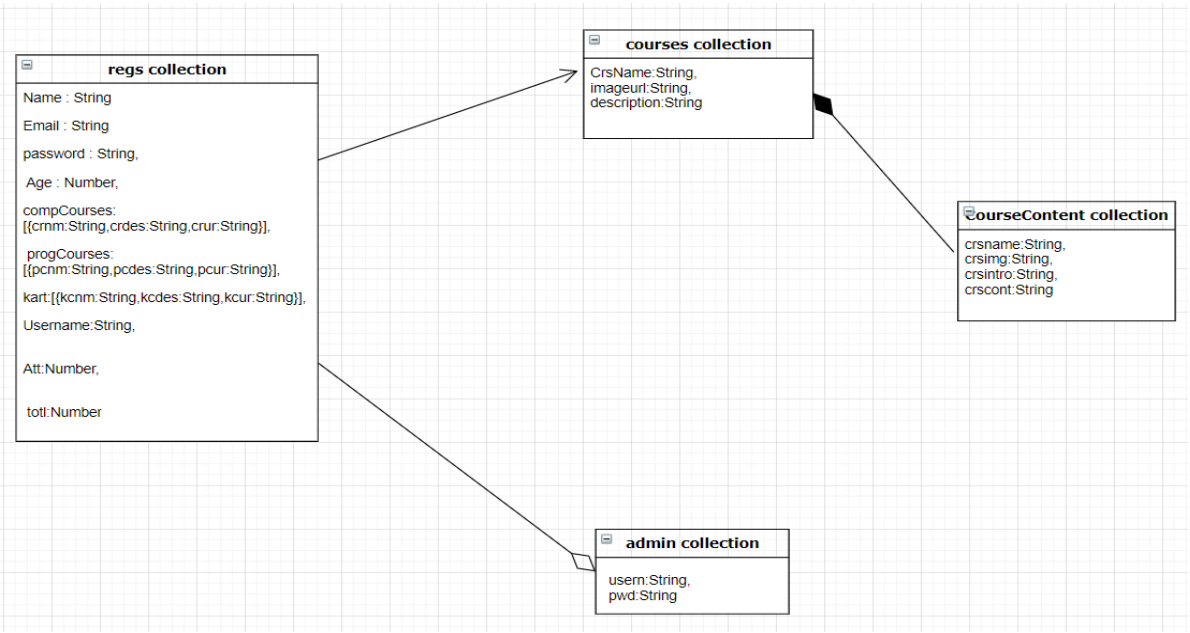
3.3 **UML DIAGRAM**

USE CASE DIAGRAM



4.**RESULTS**

4.1Database Schema



4.2 **Table Schema**

regs collection

```
{
   Name : String,
   Email : String ,
   password : String,
   Age : Number,
   compCourses:[{crnm:String,crdes:String,crur:String}],
   progCourses:[{pcnm:String,pcdes:String,pcur:String}],
   kart:[{kcnm:String,kcdes:String,kcur:String}],
   Username:String,
   Att:Number,
   totl:Number
}
```

courses collection

```
{
   CrsName:String,
   imageurl:String,
   description:String
}
```

CourseContent collection

```
{
   crsname:String,
   crsimg:String,
   crsintro:String,
   crscont:String
}
```

admin collection

{

usern:String,

pwd:String

}

4.3 **Screenshots with description**

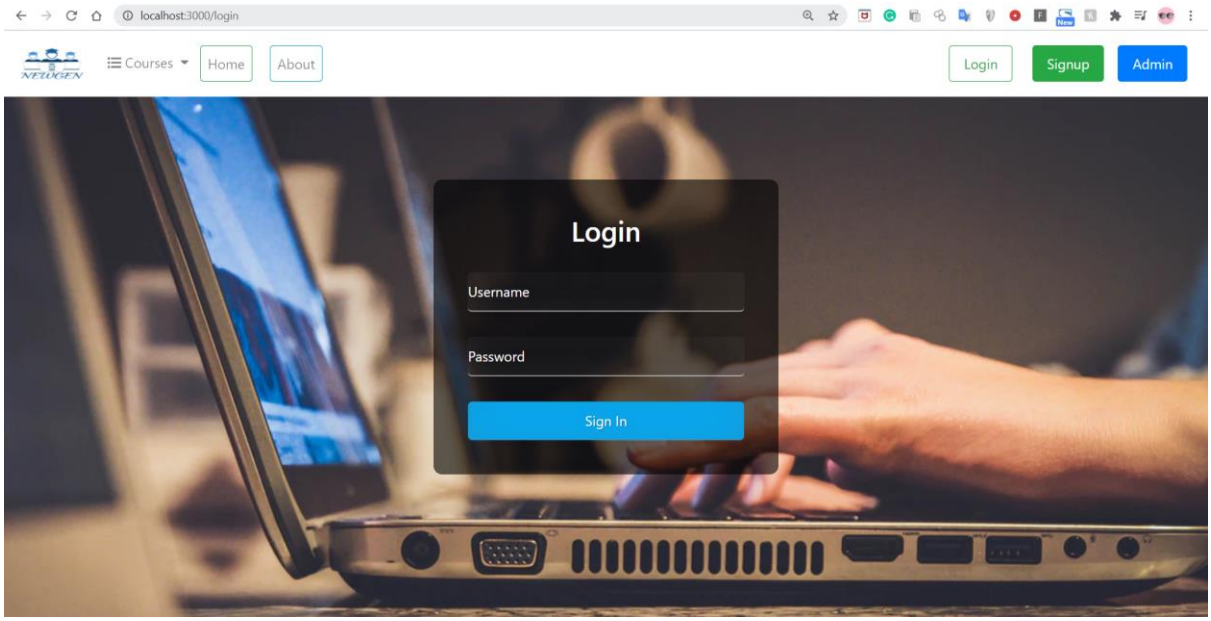


Figure: Home page

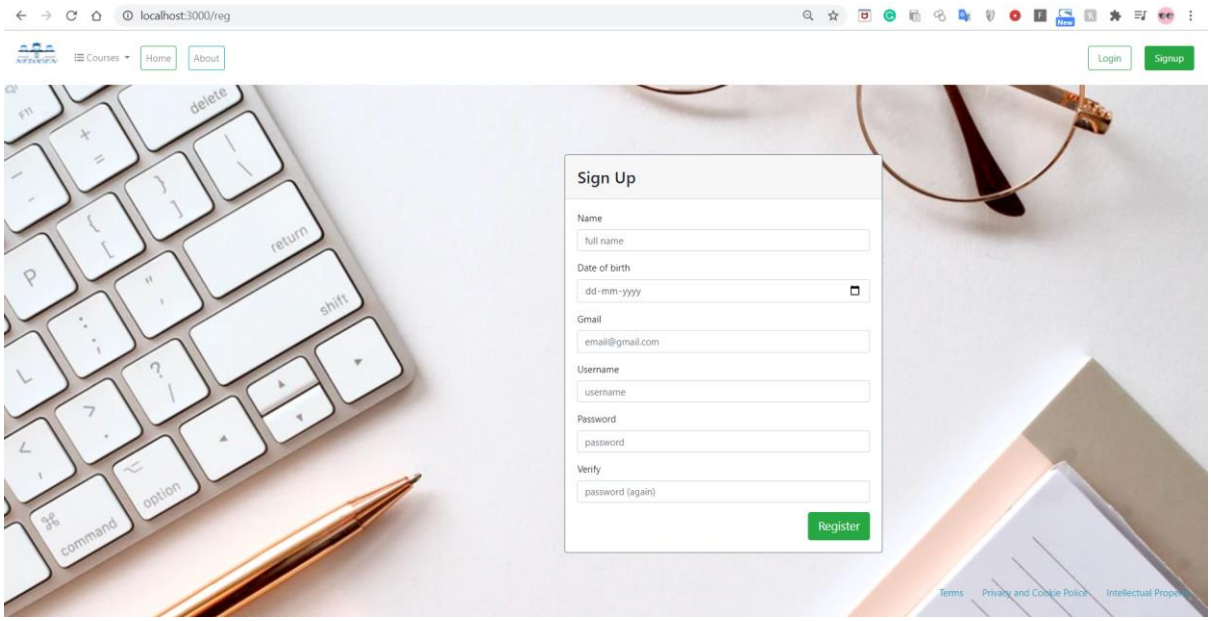


Figure: Registration page

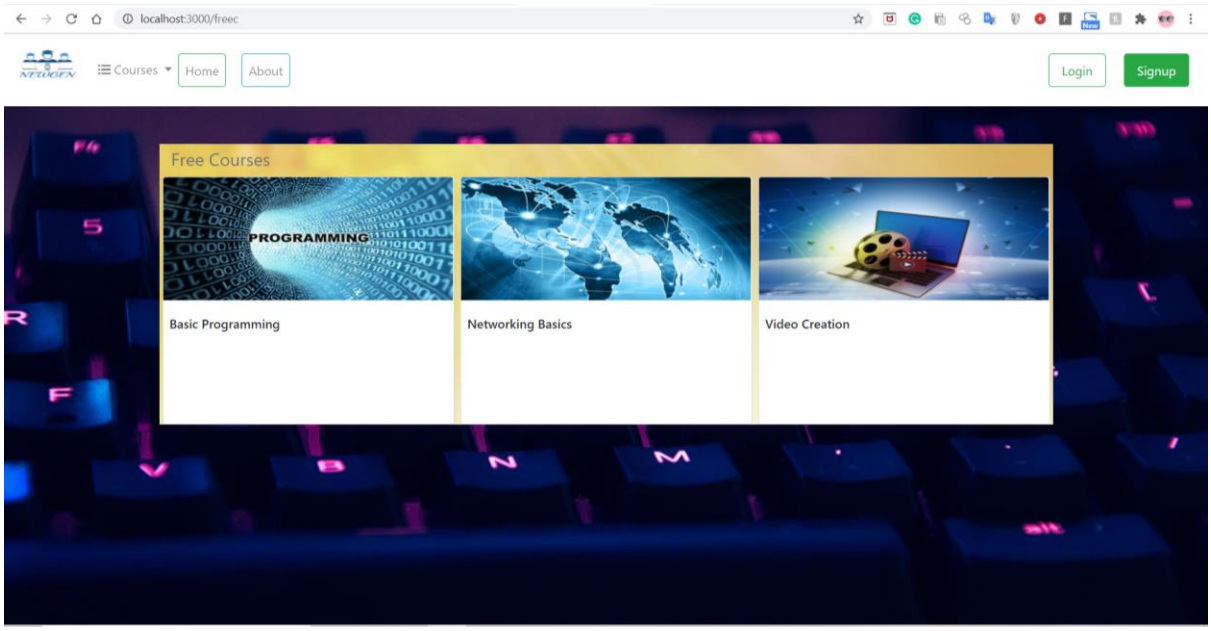


Figure: Free courses offered are displayed

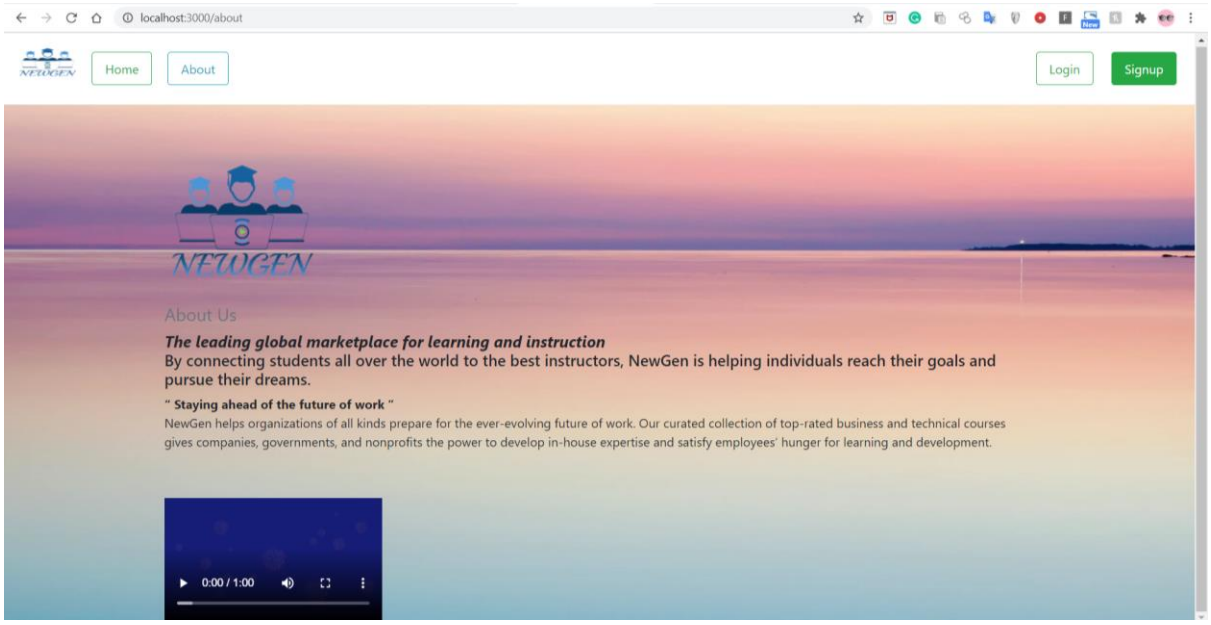


Figure : About page of the website

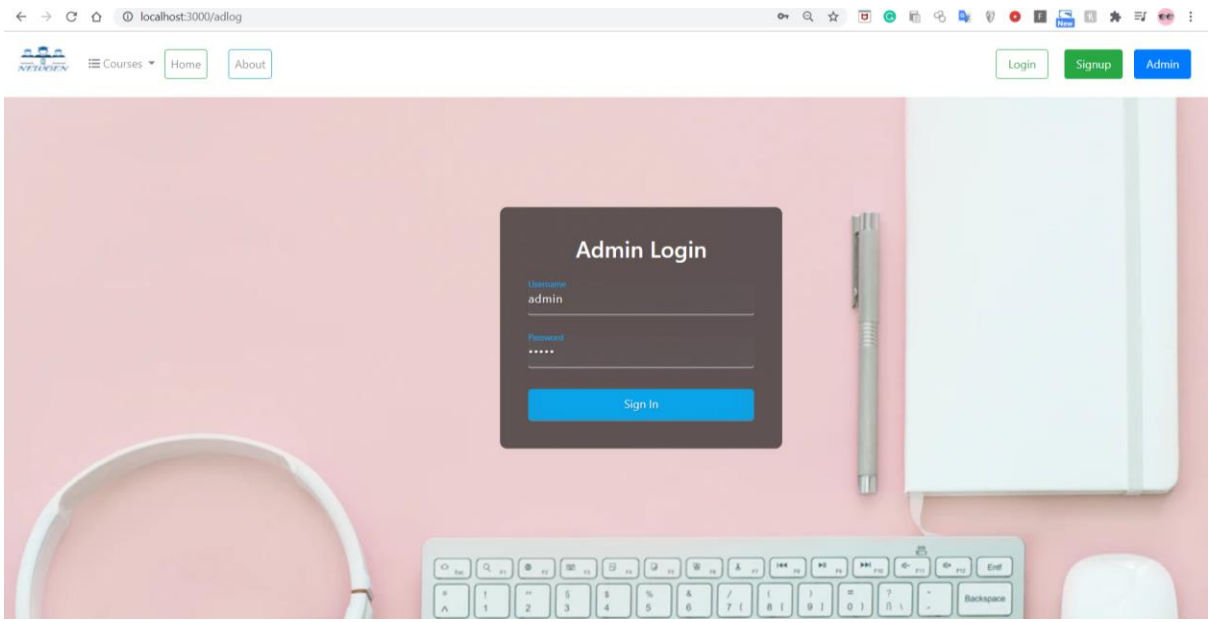


Figure : Admin Login page with credentials entered

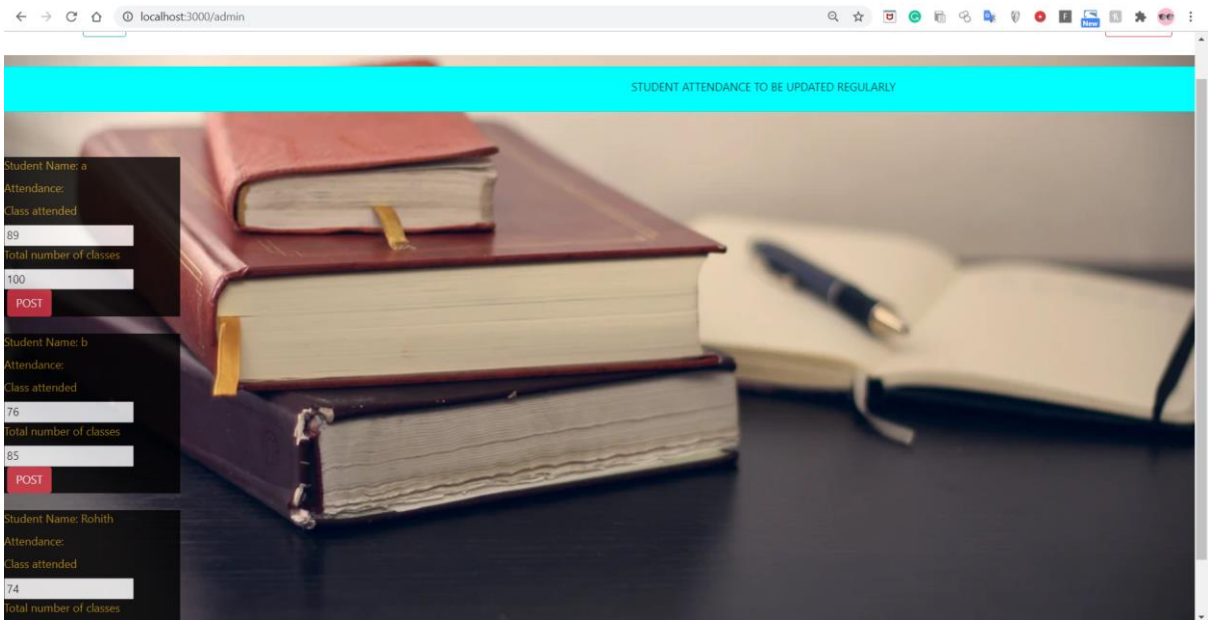


Figure: Admin portal containing the registered students attendance details and the admin can change the attendance of the student

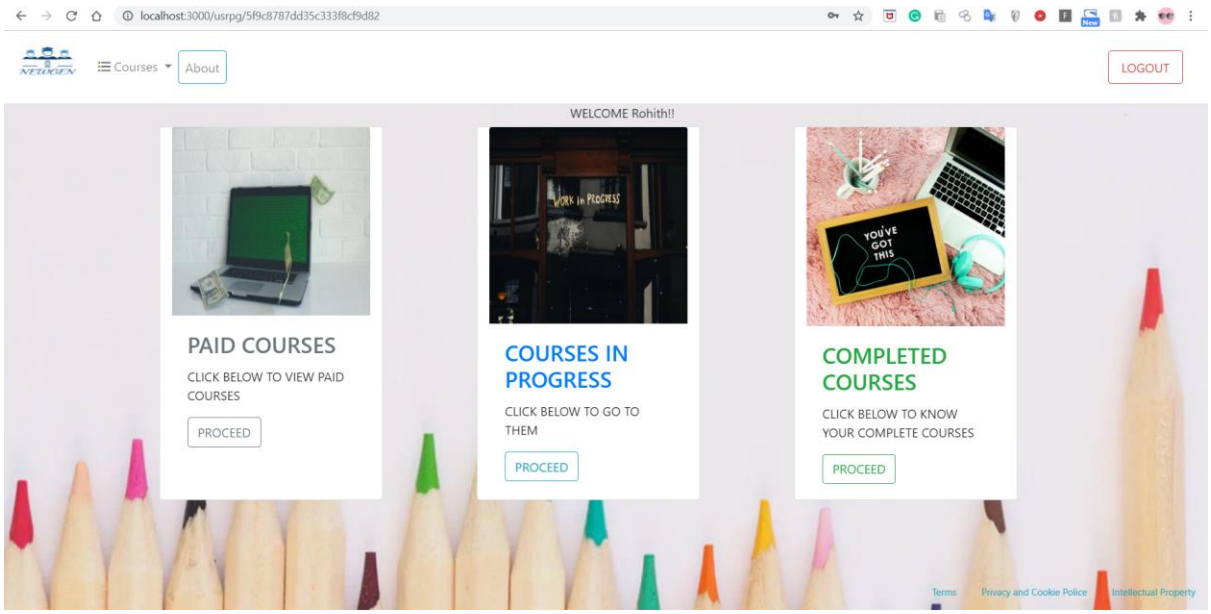


Figure: User page , when the credentials match in the login page

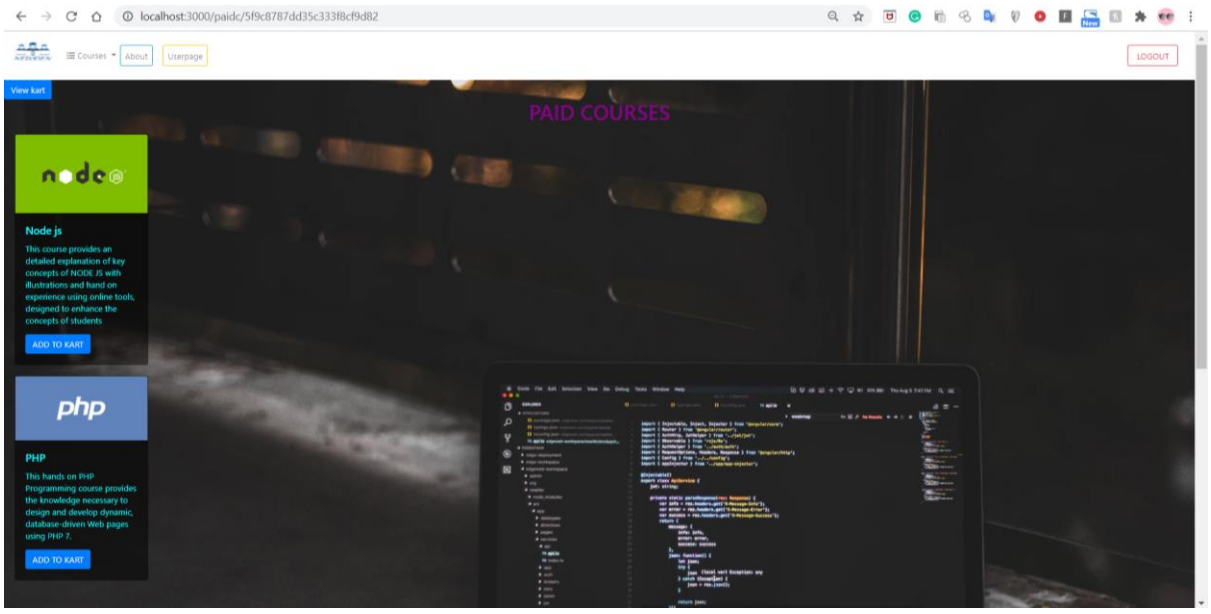


Figure: Paid courses page showing the courses available to the user

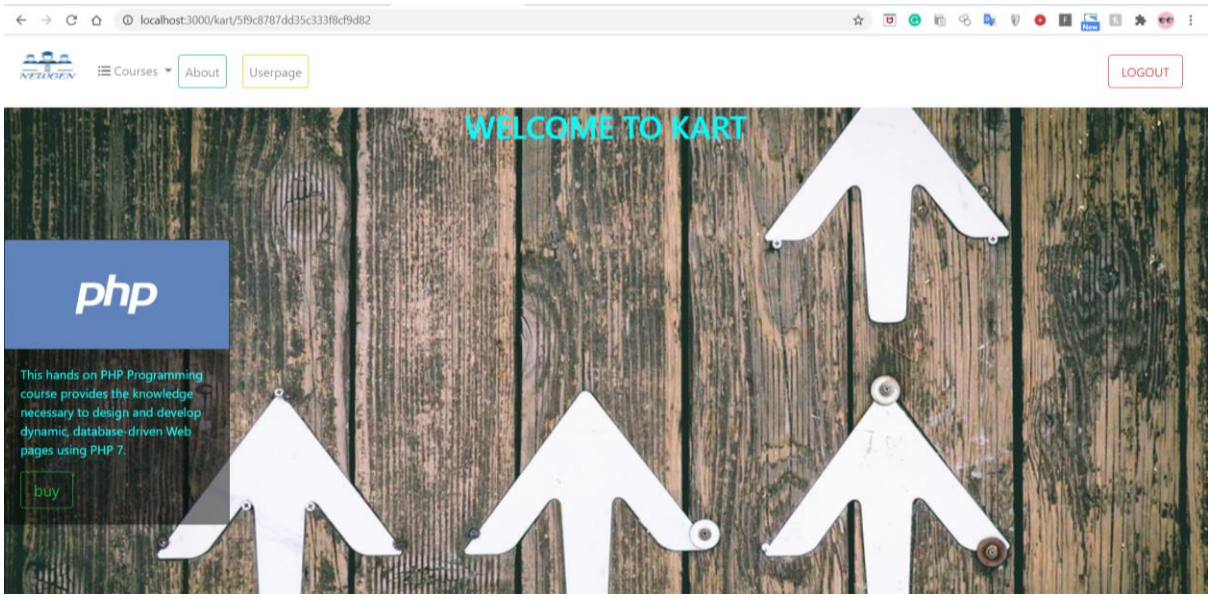


Figure: When the user clicks the add to kart option in paid courses page , the courses

are seen on the kart

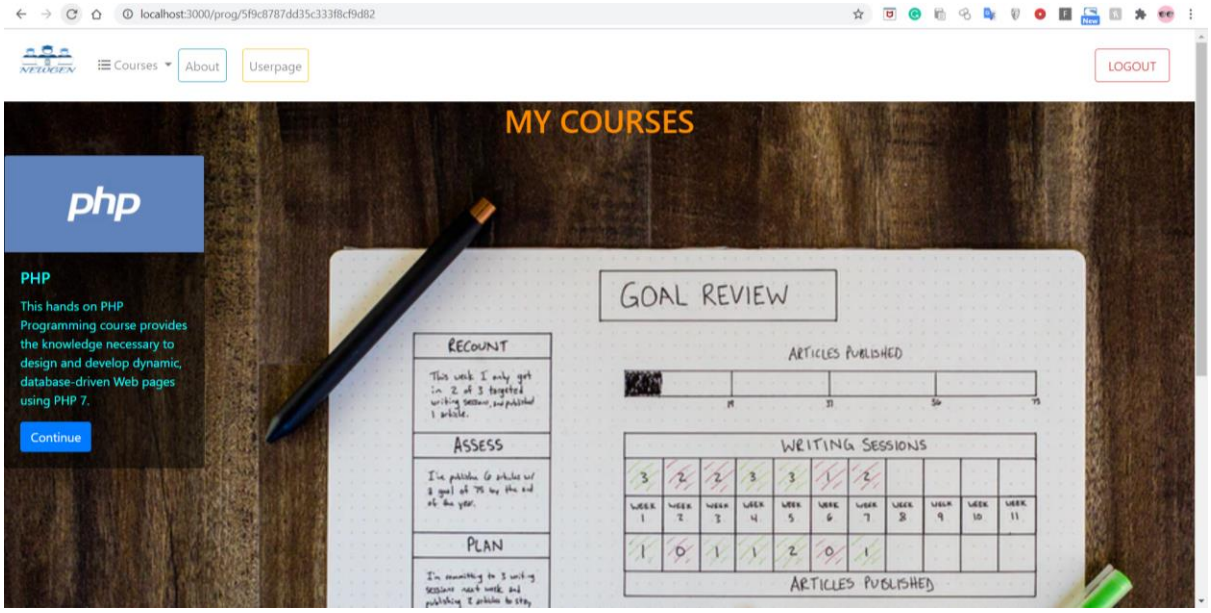


Figure: When the user clicks on buy , in the kart he can see the course in the userpage

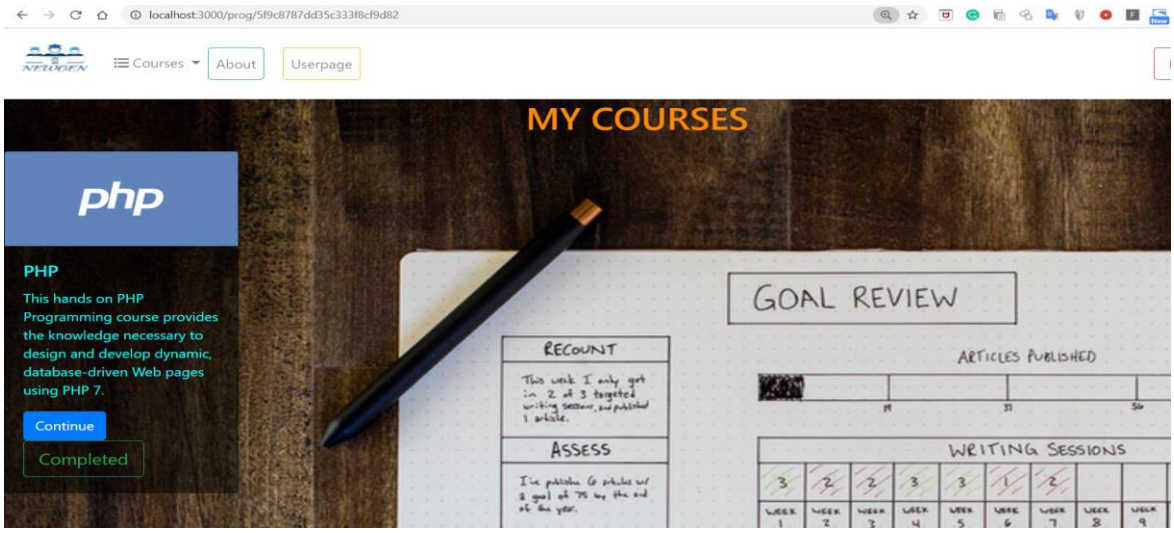


Figure: if attendance is more than 75% , then the completed course option appears as shown In image

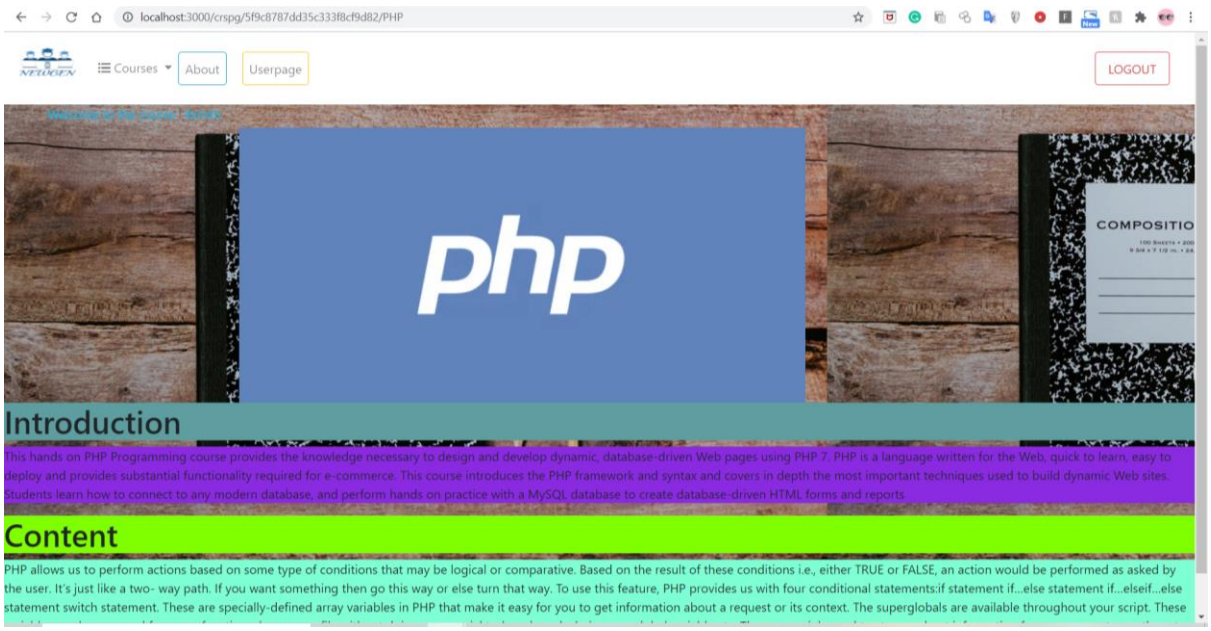


Figure: Course page

On clicking the continue option , the user is redirected to course page of the course

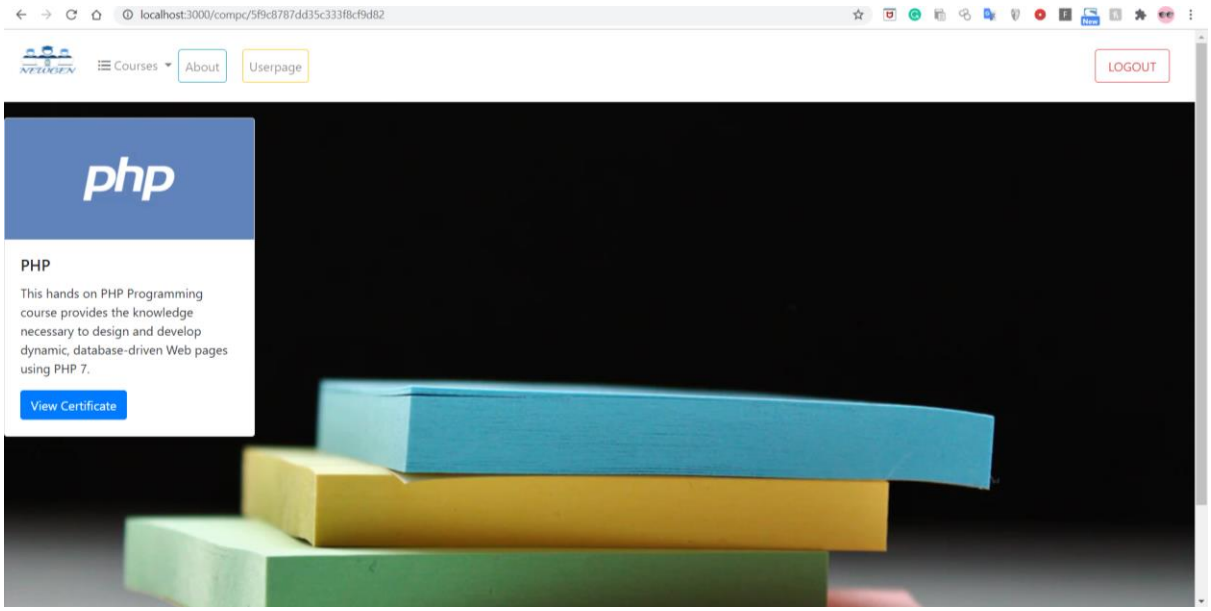


Figure: Completed Courses page

If the user selects completed option seen in my courses page , then course is added in completed courses page and the certificate is ready

Figure : Certificate generated

User then on can download the certificate or mail it to email they chose while registering for the website

5. **Conclusion**

In the Course of our project we were successfully built a online learning market place which is user friendly. The portal provides the user once registered , the choice of buying a course that is offered.

User doesn't need to make a quick choice , he can add the course to the kart first , which can be viewed later , for his reference and buy the course , view its contents and then once the attendance is updated by admin and it reaches an required percent , they can generate a certificate of completion and we provide them the option to download the certificate and mail it to their email id which they have used while registering for the portal. We made an clean user interface and abstract one and to ensure authenticity of completion of control we give the admin the controlling criteria for enabling the certificate of completion generation.

6. **References & Websites**

https://www.biztechcs.com

https://www.researchgate.net/publication/267391712_Step_by_step_building_an_e-learning_project

https://www.dotsquares.com/solutions/web-application-development/e-learning/

https://www.webcodegeeks.com/wp-content/uploads/2015/10/Building-web-apps-with-Node.js.pdf

https://stackoverflow.com/

https://www.w3resource.com/node.js/node.js-tutorials.php

7.**SAMPLE CODE**

Main Code

CODE:

```
var express = require('express');
var mongoose = require('mongoose')
//mongoose.connect("mongodb://localhost/dbk9")

mongoose.connect('mongodb+srv://rohit123:pwd123@cluster0.b8bzw.mongodb.net/prj
?retryWrites=true&w=majority', { useNewUrlParser: true, useUnifiedTopology: tr
ue, useCreateIndex: true }).then(() => console.log('MongoDB Atlas connected...
')).catch((err) => console.log(err));
var app=express();
var bodyparser=require('body-parser')
var metr = require('method-override')
app.use(bodyparser.urlencoded({extended:true}))
app.set("view engine","ejs")
mongoose.set('useFindAndModify', false);
app.use(express.static("public"))
app.use(metr("_method"));
var courses = require('./models/courses')
var regs = require('./models/regs')
var crscon = require('./models/coursecontent')
var admin = require('./models/admin')
const nodemailer = require('nodemailer');



var transporter = nodemailer.createTransport({
    service: 'Gmail',
    auth: {
        user: 'rohithkrishna.vit2000@gmail.com',
        pass: 'ldtcgkgdgsvnbsig'
    }
});
var pt ='D://5thsem//certificates//'
app.post("/mail/:id/:cn", function(req,res){
    regs.findById(req.params.id, function(err,f){
        if(err){
            console.log(err)
        }
courses.findOne({CrsName:req.params.cn}, function(err1,f1){


    console.log('Mail sent');
    transporter.sendMail({
    from: 'rohithkrishna.vit2000@gmail.com',
      to: f.Email,
```

```
      subject: f1.CrsName+"-certificate",
      text: 'Thank you for completing the course '+ f.Name,
      attachments:[
          {
              path:pt+f.Name+f1.CrsName+".pdf"
          }
      ]
    });




})


res.redirect("/usrpg/"+f._id)
    })


})
app.get("/hmp", function(req,res){

res.render("index1")

})


app.get("/login", function(req,res){
    res.render("login")
})

app.get("/reg", function(req,res){
    res.render("reg")
})


app.post("/hmp", function(req,res){
    var nm = req.body.inputName;
    var email= req.body.inputEmail3;
    var password= req.body.inputPassword3
    var age = req.body.age;
    var usrn = req.body.username;
    var regtd = {Name: nm, Email:email,password:password,Age:age,Username:usrn
};
    regs.create(regtd, function(err,f){
        if(err){
            res.send(err)
        }
        else{ console.log(f);
```

```
        res.redirect("/login")
        }
    })


})


app.get("/prog/:id",(req,res)=>{
    regs.findById(req.params.id,(err,f)=>{
        if(err){
            res.send(err)
        }
        else{
            res.render("prog",{usr:f})
        }
    })
})
app.get("/cert/:id/:crnm", function(req,res){
    var id =req.params.id;
    var cn= req.params.crnm;
    regs.findById(id,function(err,f){

        if(err){
            console.log(err)
        }
        else{

        courses.findOne({CrsName:cn}, function(err1,f1){
            if(err1){
                console.log(err1)
            }
            else{
                res.render("cert",{usr:f,crs:f1})
            }
        })

        }
    })
})


app.get("/compc/:id",function(req,res){
    regs.findById(req.params.id,function(err,f){
        if(err){
            res.send("error")

        }
        else{
```

```
            res.render("compc",{usr:f})
        }
    })
})

app.put("/update/:id/:cn",function(req,res){
courses.findOne({CrsName:req.params.cn},function(err,f){
    if(err){res.send("error")}
    else{  var de=f.description;
        var ur=f.imageurl;
        var nml=f.CrsName;
        var upd={pcnm:nml,pcdes:de,pcur:ur};
        regs.findByIdAndUpdate(req.params.id,{$push:{progCourses:upd}},functio
n(err1,f1){
if(err1){
    res.send("error")
}
else{
    res.redirect("/prog/"+f1._id);
}
 })
    }
})

})


app.put("/kartupd/:id/:cid", function(req,res){
courses.findById(req.params.cid, function(err,f){
if(err){
    res.send(err)
}
else{
var cnm = f.CrsName;
var ur= f.imageurl;
var dc = f.description;
var upd ={ kcnm:cnm,kcdes:dc,kcur:ur};
regs.findByIdAndUpdate(req.params.id, {$push:{kart:upd}}, function(err1,f1){
    if(err1){
        res.send(err1)
    }
    else{
        res.redirect("/paidc/"+f1._id)
    }
})
}

})
```

```
})



app.get("/kart/:id",function(req,res){
regs.findById(req.params.id,function(err,f){
    if(err){console.log(err)}
    else{
     res.render("kart",{usr:f})


    }
})
})


app.get("/paidc/:id",function(req,res){
    regs.findById(req.params.id,function(err,f){
        if(err){
            res.send("error!not found")
        }
        else{
            courses.find({},function(err1,f1){
         if(err1){console.log(err)}
         else{
             res.render("paidc",{usr:f,crs:f1})
         }

        })

        }
    })
})
app.get("/usrpg/:id",function(req,res){
    regs.findById(req.params.id,function(err,f){
        if(err){
            res.send(err)
        }
        else{
            res.render("userpg",{usr:f})
        }
    })
})
app.post("/login",function(req,res){
    var nm = req.body.nm;
    var pwd = req.body.pw;
    regs.findOne({Username:nm,password:pwd},function(err,f){
        if(err){res.send("Sorry these credentials are invalid")
```

```
                    res.redirect("/login")}
        else{
            res.redirect("/usrpg/"+f._id)
        }
    })
})



app.put("/compcd/:id/:cn", function(req,res){
    courses.findOne({CrsName:req.params.cn}, function(err,f){
        if(err){
            console.log(err)
        }
        else{
            var nm = f.CrsName;
            var dc = f.description;
            var ur = f.imageurl;
            var cpd= {crnm:nm,crdes:dc,crur:ur};
            regs.findByIdAndUpdate(req.params.id,{$push:{compCourses:cpd}},fun
ction(err1,f1){
                if(err1){console.log(err1)}
                else{
                    res.redirect("/compc/"+f1._id)
                }
            })

        }
    })
    })

app.get("/crspg/:id/:cn", function(req,res){
regs.findById(req.params.id,function(err,f){
    if(err){
        console.log(err)
    }
    else{
        crscon.findOne({crsname:req.params.cn}, function(err1,f1){
            if(err1){
                console.log(err1)
            }
            else{
                res.render("crspg",{usr:f,crs:f1})
            }
        })
    }
})
})
```

```
app.get("/about",(req,res)=>{
    res.render("about")
})

app.get("/about/:id",(req,res)=>{
    regs.findById(req.params.id,function(err,f){
        if(err){
            res.send(err);
        }
        else{
            res.render("abouti",{usr:f})
        }
    })
})

app.get("/adlog", function(req,res){
    res.render("adlog")
})
app.post("/adlog", function(req,res){
    var un = req.body.uname;
    var pw = req.body.pwd

    admin.findOne({usern:un,pwd:pw}, function(err,f){
        if(err){
            res.send(err)
        }
        else{
            res.redirect("/admin")
        }
    })
})

app.get("/admin", function(req,res){
    regs.find({}, function(err,f){
        if(err){res.send(err)}
        else{
            res.render("admin",{ad:f})
        }
    })
})

app.put("/attup/:id", function(req,res){
    regs.findById(req.params.id, function(err,f){
        if(err){
            console.log(err)
        }
     else{
         var tot = req.body.tot;
```

```
        var att = req.body.Att;
        regs.findByIdAndUpdate(req.params.id,{$set:{Att:att,totl:tot}}, funct
ion(err1,f1){
            if(err1){
                console.log(err1)
            }
            else{
                res.redirect("/admin")
            }
        })
     }

    })
})
app.get("/freec",(req,res)=>{
    res.render("freecourses")
})

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

app.get("/newc",(req,res)=>{
    res.render("new")
})

app.get("/new1c",(req,res)=>{
    res.render("new1")
})

app.listen(3000, function(){
    console.log("server has started")

})
```

Explanation:

We import required libraries like **express, body-parser**,**method-override**, **mongoose** and **nodemailer**. Then we import schemas of collections **regs,courses,CourseContent,Admin**. We then establish a connection with online mongo db client ,which we use to make a database and store collections in it .

The **app.listen()** function is used to bind and listen the connections on the specified host and port, then once the server starts running , the user can access the home page using url http://localhost:3000/hmp (it is run on local server of the system) , then user can access the home page of the website , from where user can navigate as per his requirements.He can

choose to register , login(if account exists) and goto admin page if he is a admin and do required operations

. Upon sending the get requests , the respective pages are rendered on the web page of the browser and and post requests sent , changes are made to database as required and method-override library was imported to make **use** of HTTP verbs such as PUT or DELETE in places where the client doesn't support it.