**Project 01 - Create your own Learning Management System**

## Initialize the Application

The version of Node in the local machine is

*v16.13.0*

The version of npm (node package manager) in the local machine is

*8.3.0*

Initialize the application by typing the following command in the terminal

*npm init --yes*

install the required packages (latest) by typing the following commands in the terminal

*npm install mongodb*

*npm i express*

*npm i mocha*

*npm i nodemon*

nodemon automatically incorporate changes to js files without the need to stop and restart the server each time a change is made to the files

add *"start": "nodemon app.js"* to *“scripts”* section of package.json

|  |
| --- |
| package.json |
| {  "name": "lms",  "version": "1.0.0",  "description": "",  "main": "index.js",  "scripts": {  "test": "echo \"Error: no test specified\" && exit 1",  "start": "nodemon app.js"  },  "keywords": [],  "author": "",  "license": "ISC",  "dependencies": {  "express": "4.17.2",  "mocha": "9.2.0",  "mongodb": "4.3.1",  "nodemon": "2.0.15"  }  } |

Kindly note the carat ‘^’ sign is removed before the version numbers so that the version is locked to avoid version mismatch errors in case project is shared between teams.

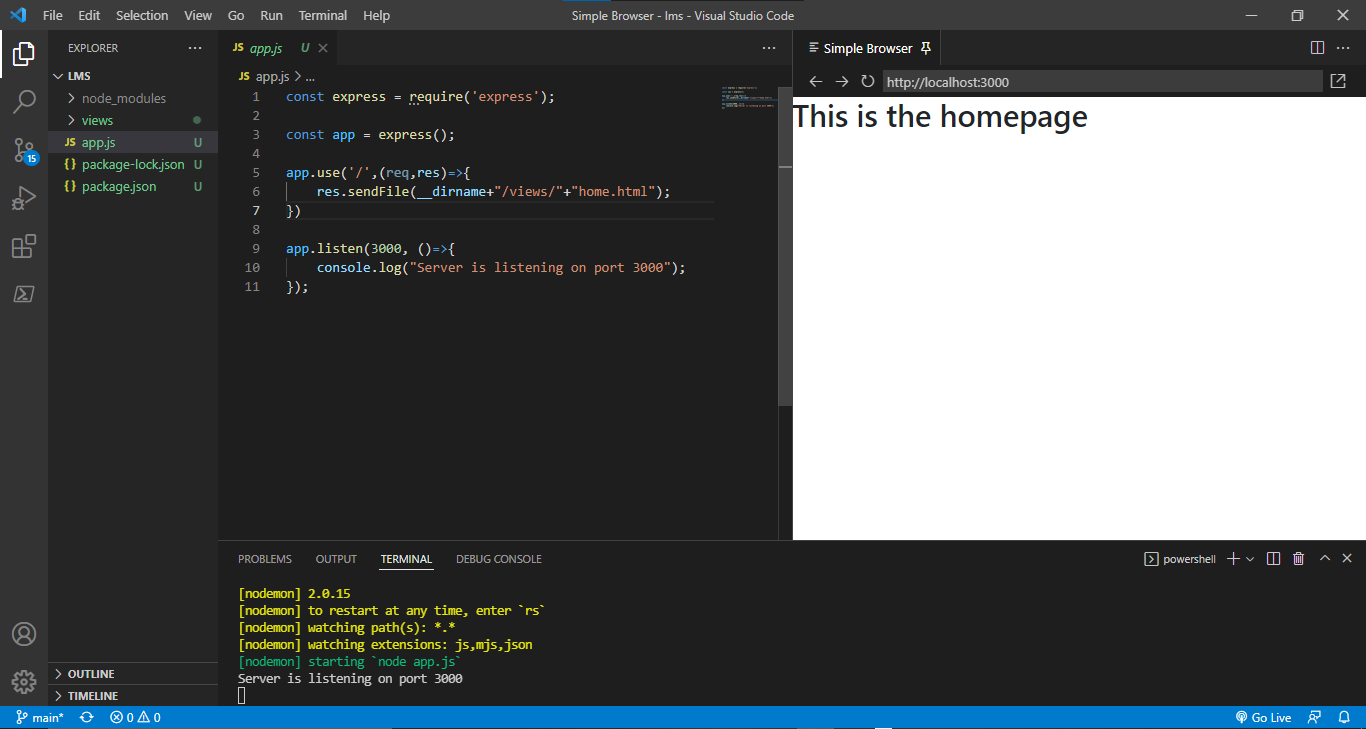
Start Express on localhost:3000

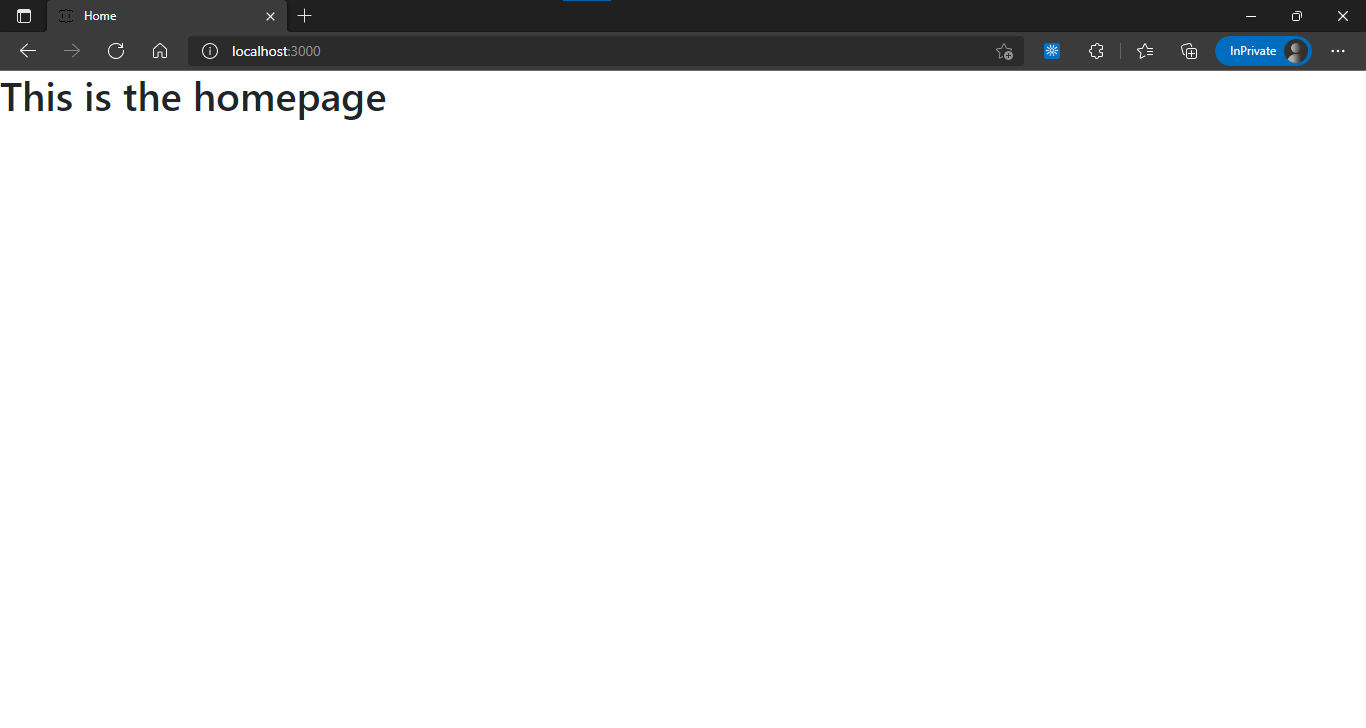
|  |
| --- |
| app.js |
| const express = require('express');  const app = express();  app.use('/',(req,res)=>{  res.sendFile(\_\_dirname+"/views/"+"home.html");  })  app.listen(3000, ()=>{  console.log("Server is listening on port 3000");  }); |
| home.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Home</title>  </head>  <body>  <h1>This is the homepage</h1>  </body>  </html> |

Start express by typing the following command in the terminal

*npm start*

the homepage will be displayed in the browser at http://localhost:3000





## Setup the routes

Setup routing for user and course. The user route will contain all the routes for views like registration, login, edit user and navbar and the course route will contain all the routes for views like course update, course delete and dashboard. The routing will be added to app.js as middleware.

|  |
| --- |
| user\_routes.js |
| const express = require('express');  const path = require('path');  const router = express.Router();  router.get('/home', (req,res)=>{  res.sendFile(path.join(\_\_dirname,'../','views','home.html'));  });  router.get('/navbar', (req,res)=>{  res.sendFile(path.join(\_\_dirname,'../','views','navbar.html'));  });  router.get('/edit', (req,res)=>{  res.sendFile(path.join(\_\_dirname,'../','views','user\_edit.html'));  });  module.exports = router; |
| course\_routes.js |
| const express = require('express');  const path = require('path');  const router = express.Router();  router.get('/', (req,res)=>{  res.sendFile(path.join(\_\_dirname,'../','views','course\_new.html'));  });  module.exports = router; |
| app.js |
| const express = require('express');  const userRoutes = require('./routes/user\_routes');  const courseRoutes = require('./routes/course\_routes');  const app = express();  app.use('/user',userRoutes);  app.use('/course',courseRoutes);  app.listen(3000, ()=>{  console.log("Server is listening on port 3000");  }); |

## Create the Models

|  |
| --- |
| Course-model.js |
| const courses = [];  module.exports = class CourseModels {  constructor(courseName,courseCategory,courseOneLiner,courseDuration,courseLanguage,courseDescription,courseLessons,courseCoverPhoto,id){  this.courseName = courseName;  this.courseCategory = courseCategory;  this.courseOneLiner = courseOneLiner;  this.courseDuration = courseDuration;  this.courseLanguage = courseLanguage;  this.courseDescription = courseDescription;  this.courseLessons = courseLessons;  this.courseCoverPhoto = courseCoverPhoto;  this.\_id = id;  }    } |
| User-model.js |
| const users = [];  module.exports = class userModels{    constructor(userName,userType,userEmail,userPassword,id){  this.userName = userName;  this.userType = userType;  this.userEmail = userEmail;  this.userPassword = userPassword;  this.\_id = id;  }  } |

## Create the Static Views

Install bootstrap using the following CDN link provided by www.bootstrapcdn.com

https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css

add the below tag to the head tag in the html file

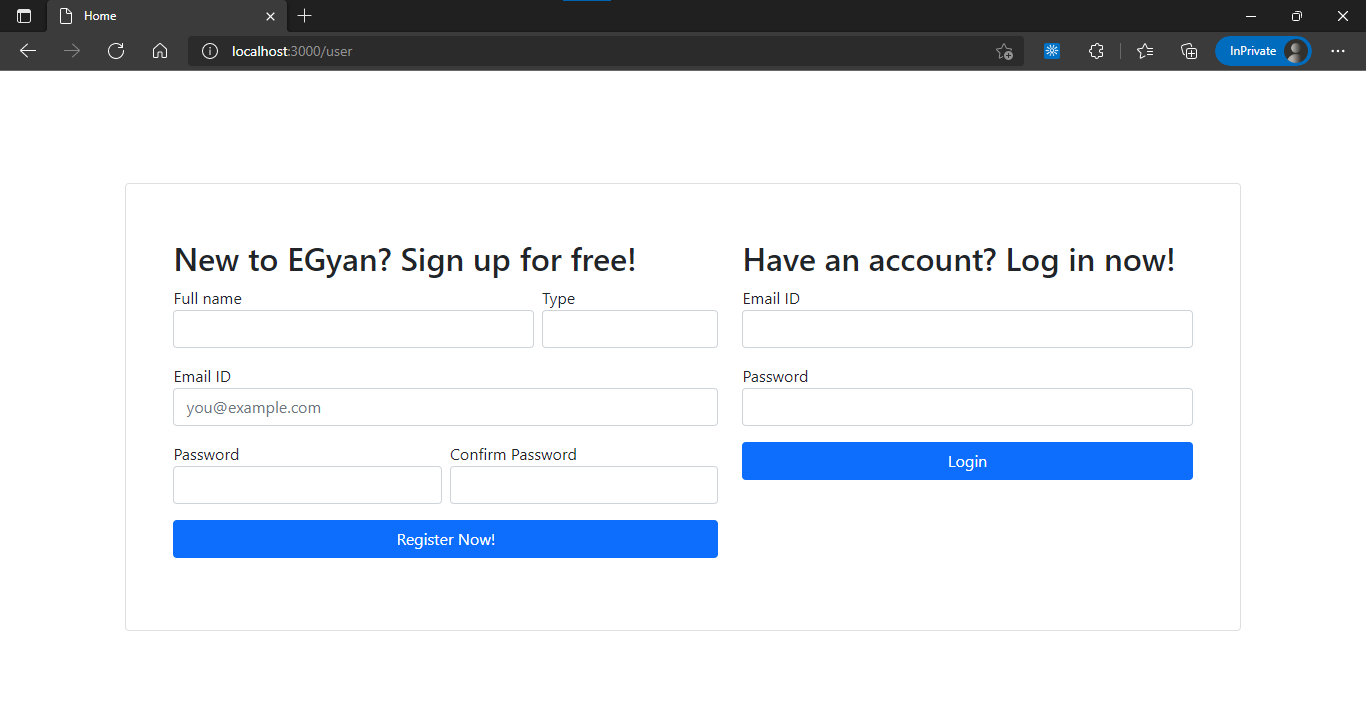
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">

Type the following command in terminal to install bootstrap

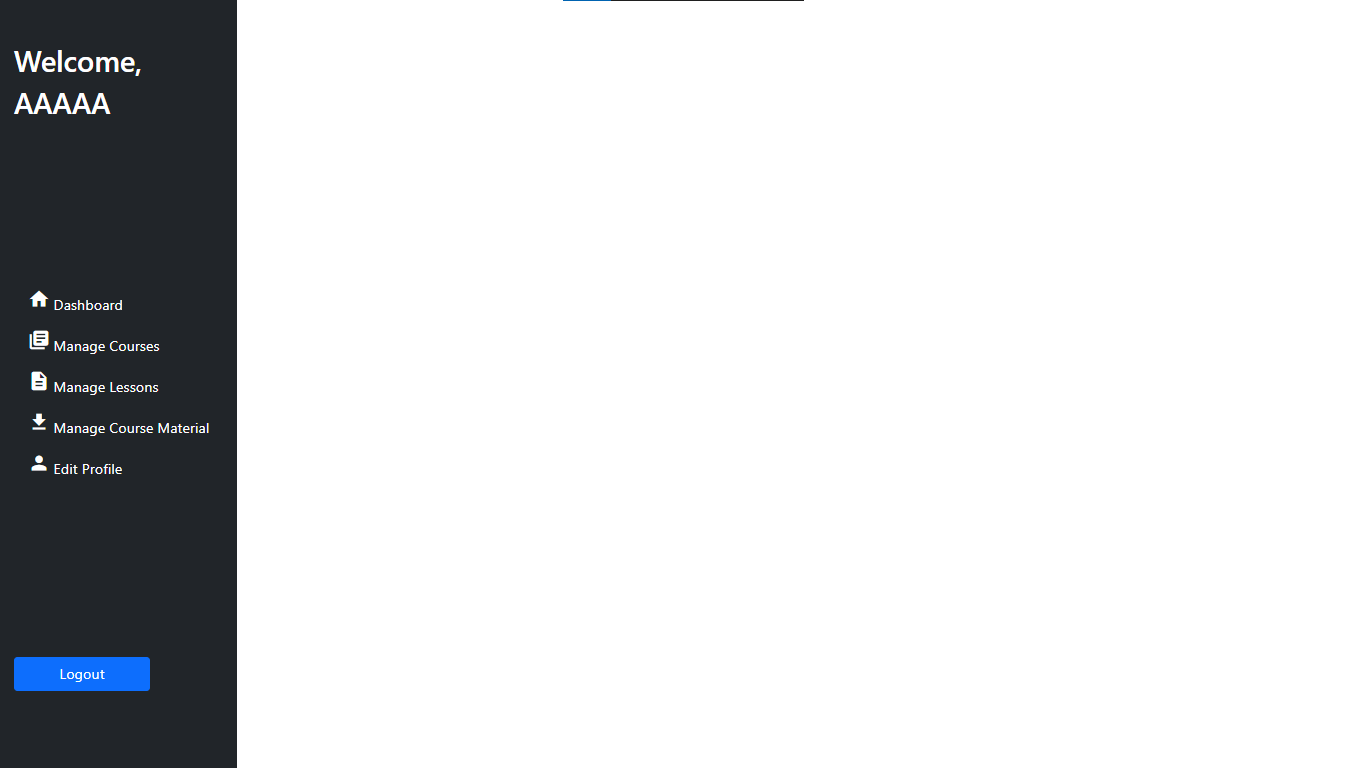
*npm i bootstrap*

Type the code to display the required forms in homepage

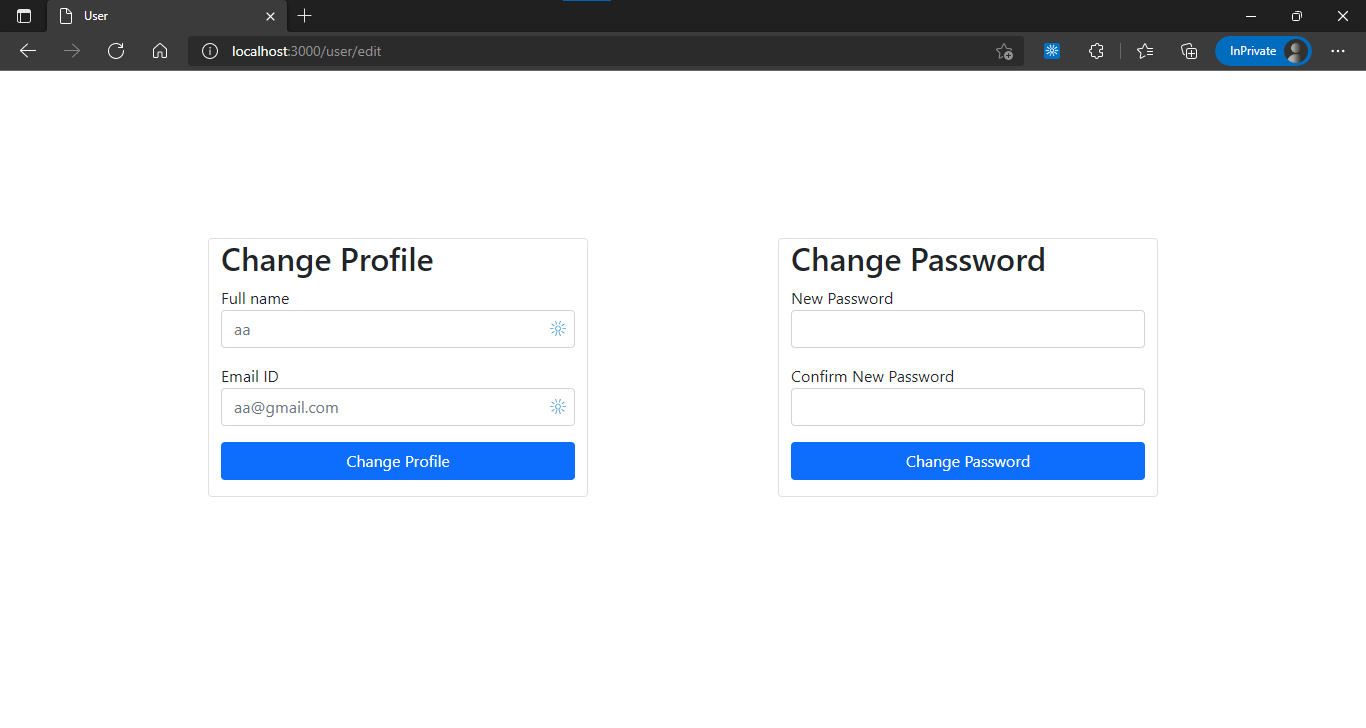
|  |
| --- |
| home.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Home</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  </head>  <body>  <div class="container">  <form action="">  <div class="card style="margin-top: 10%; padding-top: 5%; padding-bottom: 5%;">  <div class="row justify-content-center">  <div class="col-6 align-self-center">  <h2>New to EGyan? Sign up for free!</h2>  <div class="mb-3">  <div class="row g-2">  <div class="col-8">  Full name<input class="form-control" name="regName" type="text" />  </div>  <div class="col">  Type<input class="form-control" name="regType" type="text" />  </div>  </div>  </div>  <div class="mb-3">  Email ID<input class="form-control" name="regEmail" type="email" placeholder="you@example.com"/>  </div>  <div class="mb-3">  <div class="row g-2">  <div class="col">  Password<input class="form-control" name="regPassword" type="password" />  </div>  <div class="col">  Confirm Password<input class="form-control" name="regConfPassword" type="password" />  </div>  </div>  </div>  <div class="mb-3 d-grid gap-2">  <button class="btn btn-primary" type="submit" name="register">Register Now!</button>  </div>  </div>  <div class="col-5">  <h2>Have an account? Log in now!</h2>  <div class="mb-3">  Email ID<input class="form-control" name="LogEmail" type="email" />  </div>  <div class="mb-3">  Password<input class="form-control" name="LogPassword" type="password" />  </div>  <div class="mb-3 d-grid gap-2">  <button class="btn btn-primary" type="submit" name="login">Login</button>  </div>  </div>  </div>  </div>  </form>  </div>  </body>  </html> |



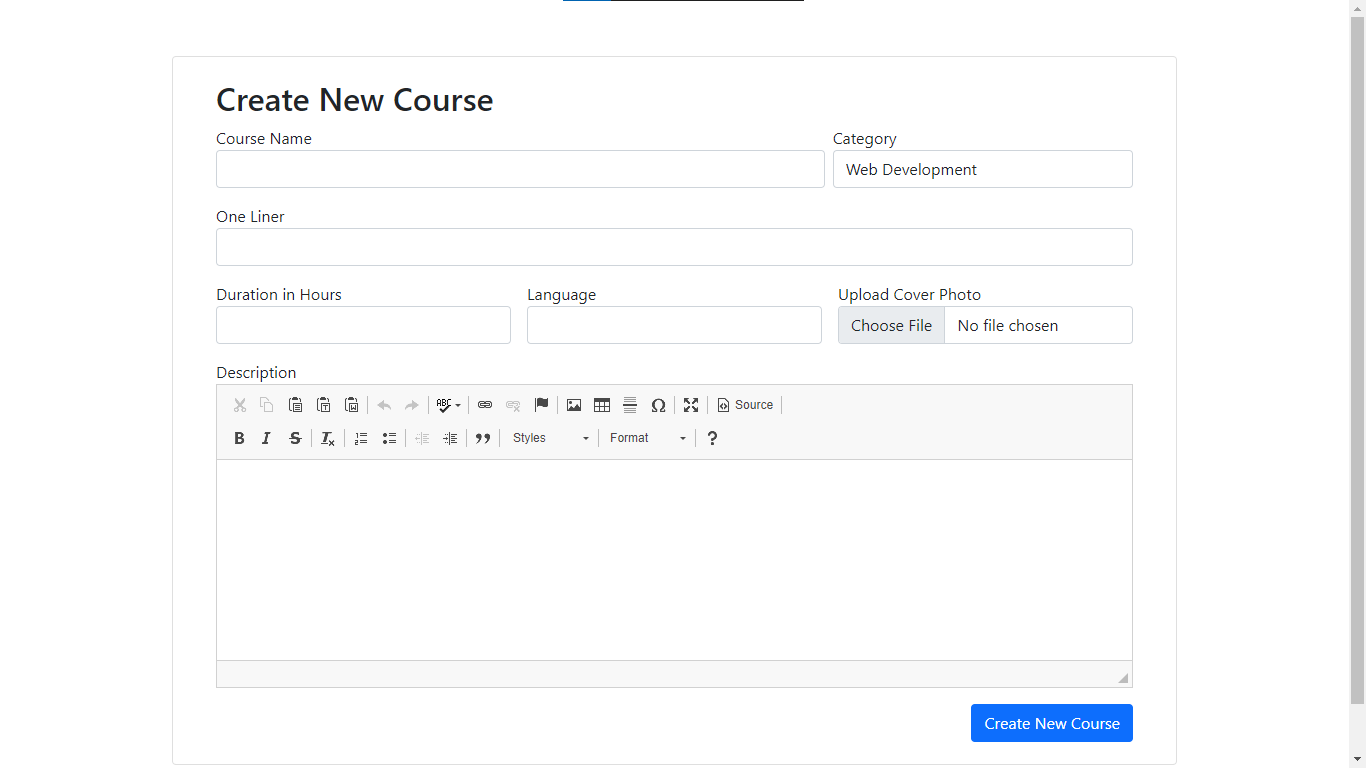
|  |
| --- |
| navbar.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Navbar</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">  </head>  <body>  <nav class="navbar navbar-dark bg-dark text-white" style="position: fixed; padding: 1%; height: 100%;">  <ul class="nav flex-column">  <div class="navbar-brand" style="position: fixed; top: 5%;">  <h2>Welcome,</h2>  <h2>AAAAA</h2>  </div>  <li class="nav-item">  <a class="nav-link text-white" href="#">  <span class="material-icons">home</span>  Dashboard  </a>  </li>  <li class="nav-item">  <a class="nav-link text-white" href="#">  <span class="material-icons">library\_books</span>  Manage Courses</a>  </li>  <li class="nav-item">  <a class="nav-link text-white" href="#">  <span class="material-icons">description</span>  Manage Lessons  </a>  </li>  <li class="nav-item">  <a class="nav-link text-white" href="#">  <span class="material-icons">download</span>  Manage Course Material  </a>  </li>  <li class="nav-item">  <a class="nav-link text-white" href="#">  <span class="material-icons">person</span>  Edit Profile  </a>  </li>  <button class="btn btn-primary" type="button" style="position: fixed; bottom: 10%; width: 10%;">Logout</button>  </ul>  </nav>  </body>  </html> |



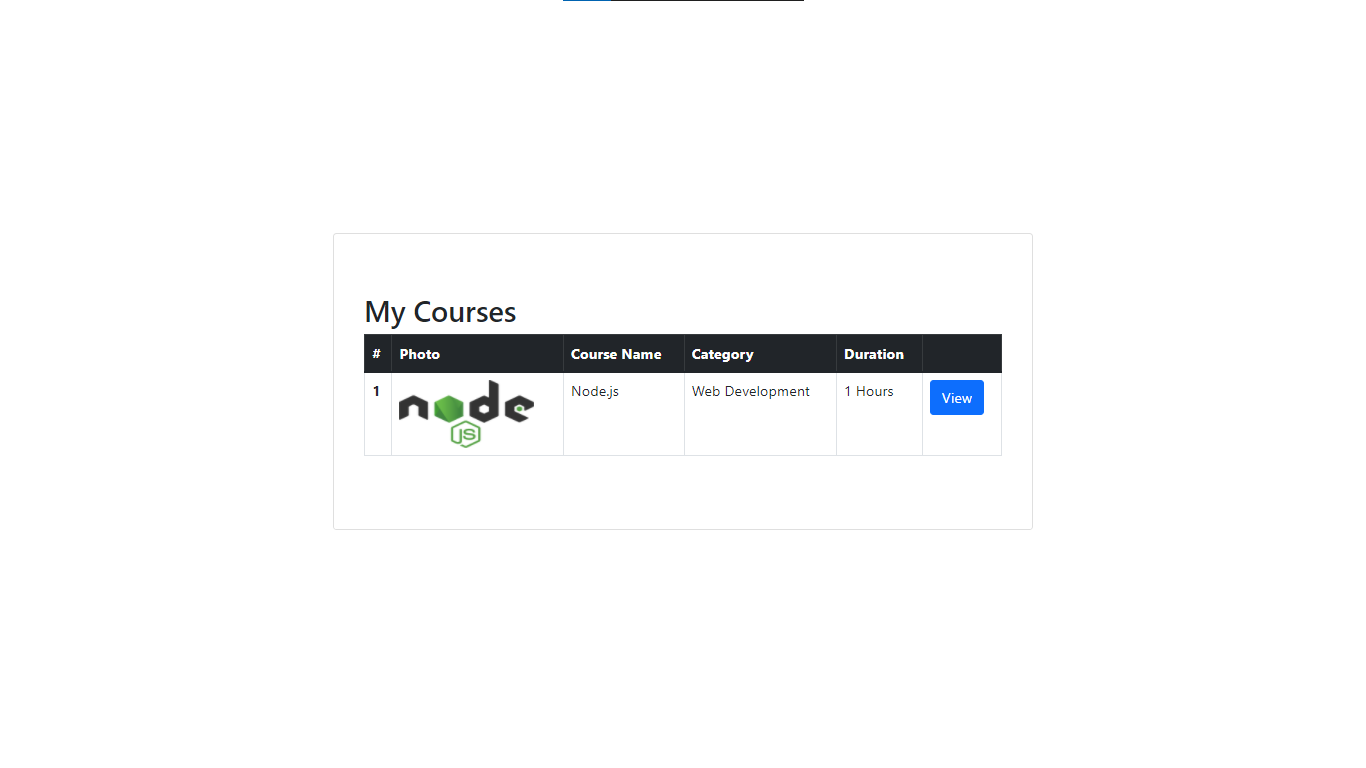
|  |
| --- |
| user\_edit.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>User</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  </head>  <body>  <div class="container">  <form action="">  <div class="row justify-content-around" style="margin-top: 10%; padding-top: 5%; padding-bottom: 5%;">  <div class="card col-4">  <h2>Change Profile</h2>  <div class="mb-3">  Full name<input class="form-control" name="regName" type="text" placeholder="aa" />  </div>  <div class="mb-3">  Email ID<input class="form-control" name="regPassword" type="password" placeholder="aa@gmail.com" />  </div>  <div class="mb-3 d-grid gap-2">  <button class="btn btn-primary" type="submit" name="changeProfile">Change Profile</button>  </div>  </div>  <div class="card col-4">  <h2>Change Password</h2>  <div class="mb-3">  New Password<input class="form-control" name="regPassword" type="password" />  </div>  <div class="mb-3">  Confirm New Password<input class="form-control" name="regConfPassword" type="password" />  </div>  <div class="mb-3 d-grid gap-2">  <button class="btn btn-primary" type="submit" name="changePassword">Change Password</button>  </div>  </div>  </div>  </form>  </div>  </body>  </html> |



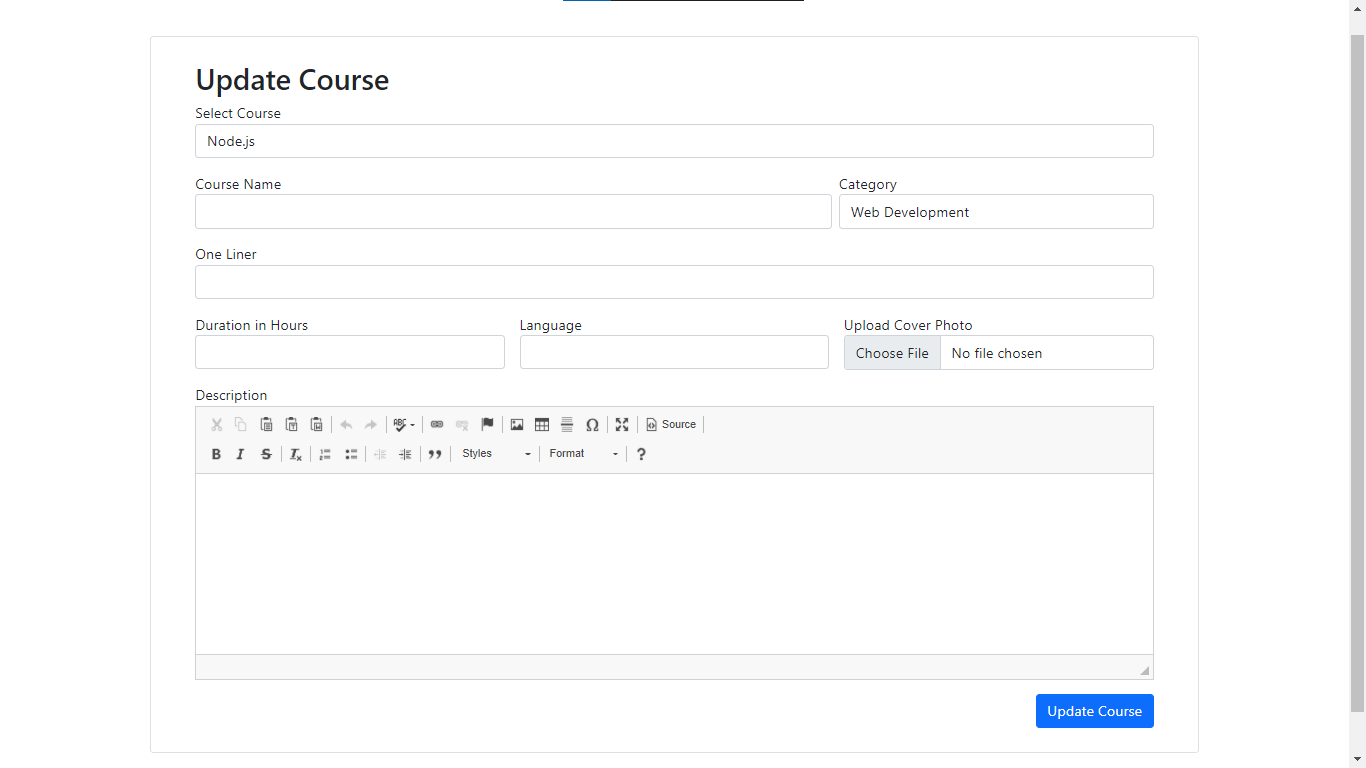
|  |
| --- |
| course\_new.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>New Course</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  <script src="//cdn.ckeditor.com/4.17.2/standard/ckeditor.js"></script>  </head>  <body>  <div class="container">  <form action="">  <div class="card" style="margin: 5%; padding-top: 2%; padding-bottom: 2%;">  <div class="row justify-content-center">  <div class="col-11">  <h2>Create New Course</h2>  <div class="mb-3">  <div class="row g-2">  <div class="col-8">  Course Name<input class="form-control" name="courseName" type="text" />  </div>  <div class="col">  Category  <select class="form-control" name="courseCategory" >  <option value="1">Web Development</option>  <option value="2">Data Science</option>  </select>  </div>  </div>  </div>  <div class="mb-3">  One Liner<input class="form-control" name="courseOneLiner" type="text" />  </div>  <div class="mb-3">  <div class="row g-3">  <div class="col">  Duration in Hours<input class="form-control" name="courseDuration" type="text" />  </div>  <div class="col">  Language<input class="form-control" name="courseLanguage" type="text" />  </div>  <div class="col">  Upload Cover Photo<input class="form-control" name="courseCoverPhoto" type="file" />  </div>  </div>  </div>  <div class="mb-3">  Description  <textarea class="form-control" name="courseDescription"></textarea>  <script type="text/javascript">  CKEDITOR.replace( 'courseDescription' );  </script>  </div>  <button class="btn btn-primary" type="submit" style="float: right;">Create New Course</button>  </div>  </div>  </div>  </form>  </div>  </body>  </html> |



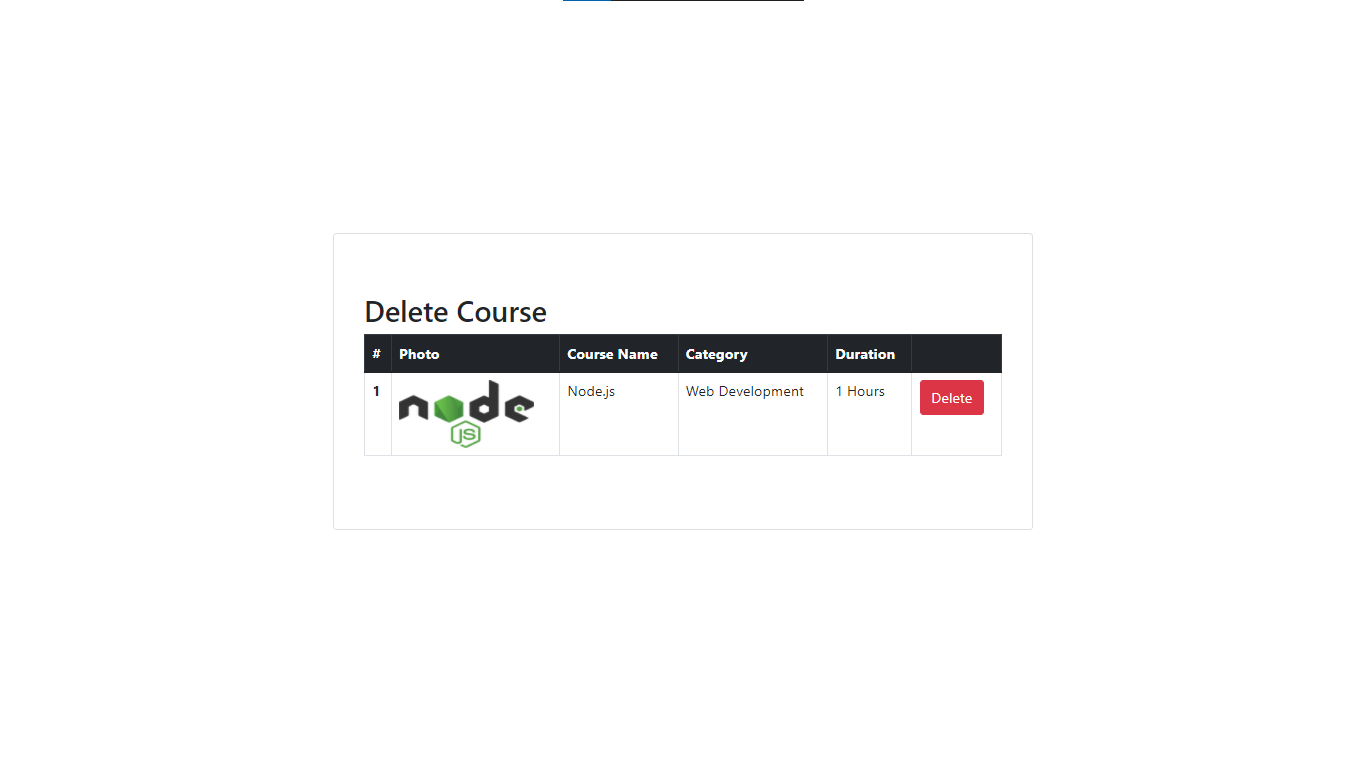
|  |
| --- |
| dashboard.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Dashboard</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  </head>  <body>  <div class="container">  <div class="card" style="margin: 20%; padding-top: 5%; padding-bottom: 5%;">  <div class="row justify-content-center">  <div class="col-11">  <h2>My Courses</h2>  <table class="table table-bordered">  <thead class="table-dark">  <tr>  <th scope="col">#</th>  <th scope="col">Photo</th>  <th scope="col">Course Name</th>  <th scope="col">Category</th>  <th scope="col">Duration</th>  <th scope="col"></th>  </tr>  </thead>  <tbody>  <tr>  <th scope="row">1</th>  <td>  <img src="https://seeklogo.com/images/N/nodejs-logo-065257DE24-seeklogo.com.png" style="height: 75px; width: 150px;">  </td>  <td>Node.js</td>  <td>Web Development</td>  <td>1 Hours</td>  <td>  <button class="btn btn-primary" type="button">View</button>  </td>  </tr>  </tbody>  </table>  </div>  </div>  </div>  </div>  </body>  </html> |



|  |
| --- |
| course\_update.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Update Course</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  <script src="//cdn.ckeditor.com/4.17.2/standard/ckeditor.js"></script>  </head>  <body>  <div class="container">  <form action="">  <div class="card" style="margin: 5%; padding-top: 2%; padding-bottom: 2%;">  <div class="row justify-content-center">  <div class="col-11">  <h2>Update Course</h2>  <div class="mb-3">  Select Course  <select class="form-control" name="courseName">  <option value="1">Node.js</option>  <option value="2">Angular.js</option>  </select>  </div>  <div class="mb-3">  <div class="row g-2">  <div class="col-8">  Course Name<input class="form-control" name="courseName" type="text" />  </div>  <div class="col">  Category  <select class="form-control" name="courseCategory">  <option value="1">Web Development</option>  <option value="2">Data Science</option>  </select>  </div>  </div>  </div>  <div class="mb-3">  One Liner<input class="form-control" name="courseOneLiner" type="text" />  </div>  <div class="mb-3">  <div class="row g-3">  <div class="col">  Duration in Hours<input class="form-control" name="courseDuration" type="text" />  </div>  <div class="col">  Language<input class="form-control" name="courseLanguage" type="text" />  </div>  <div class="col">  Upload Cover Photo<input class="form-control" name="courseCoverPhoto" type="file" />  </div>  </div>  </div>  <div class="mb-3">  Description  <textarea class="form-control" name="courseDescription"></textarea>  <script type="text/javascript">  CKEDITOR.replace( 'courseDescription' );  </script>  </div>  <button class="btn btn-primary" type="submit" style="float: right;">Update Course</button>  </div>  </div>  </div>  </form>  </div>  </body>  </html> |



|  |
| --- |
| course\_delete.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Delete Course</title>  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" crossorigin="anonymous">  </head>  <body>  <div class="container">  <div class="card" style="margin: 20%; padding-top: 5%; padding-bottom: 5%;">  <div class="row justify-content-center">  <div class="col-11">  <h2>Delete Course</h2>  <table class="table table-bordered">  <thead class="table-dark">  <tr>  <th scope="col">#</th>  <th scope="col">Photo</th>  <th scope="col">Course Name</th>  <th scope="col">Category</th>  <th scope="col">Duration</th>  <th scope="col"></th>  </tr>  </thead>  <tbody>  <tr>  <th scope="row">1</th>  <td>  <img src="https://seeklogo.com/images/N/nodejs-logo-065257DE24-seeklogo.com.png" style="height: 75px; width: 150px;">  </td>  <td>Node.js</td>  <td>Web Development</td>  <td>1 Hours</td>  <td>  <button class="btn btn-danger" type="button">Delete</button>  </td>  </tr>  </tbody>  </table>  </div>  </div>  </div>  </div>  </body>  </html> |



## Create the Dynamic Views

The views created are static. For the views to render dynamic content, a template engine I used which modifies the look and feel of the code in the HTML. Pug will be used. Most of the content is based on the content in the HTML files above, except for the starting and ending tags, pug has only a starting tag without the “<” and “>”. The format differentiates tags based on indentation.

Install pug by entering the following command in terminal

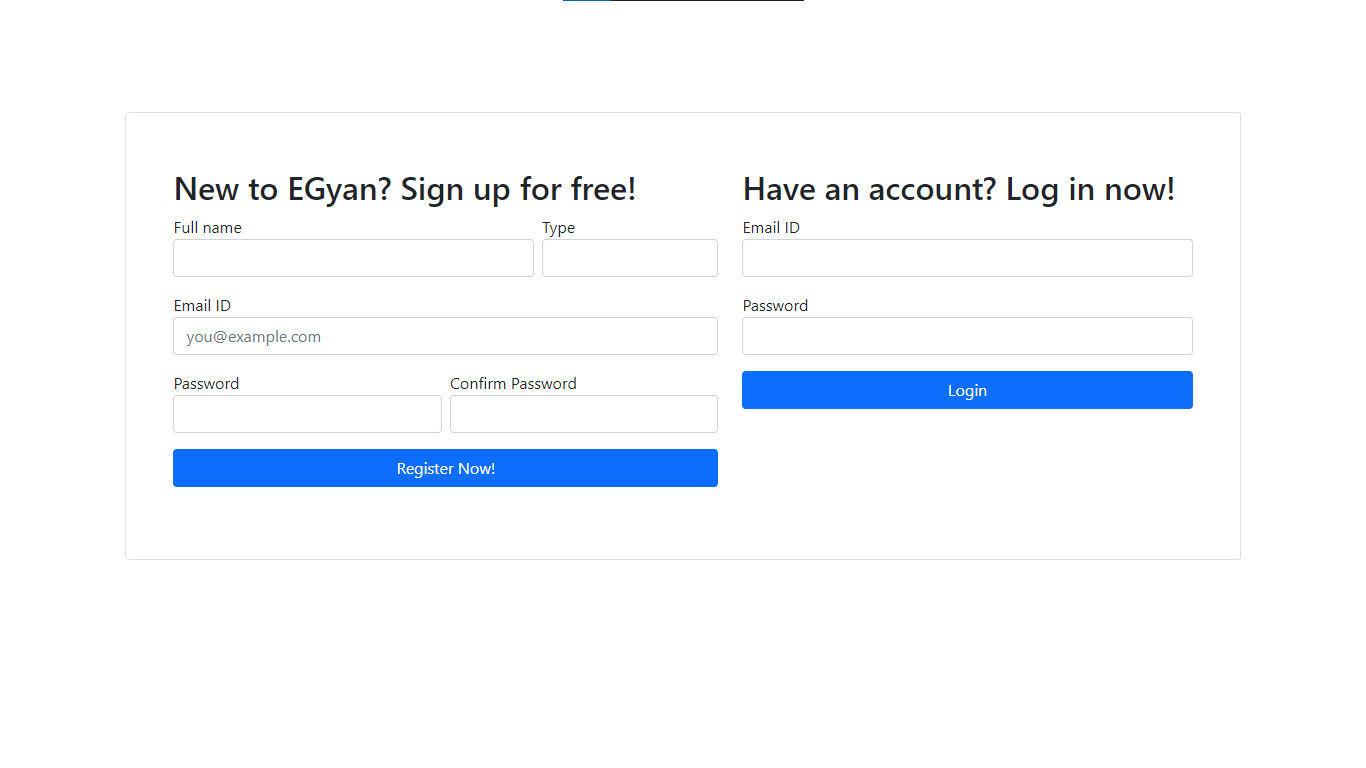
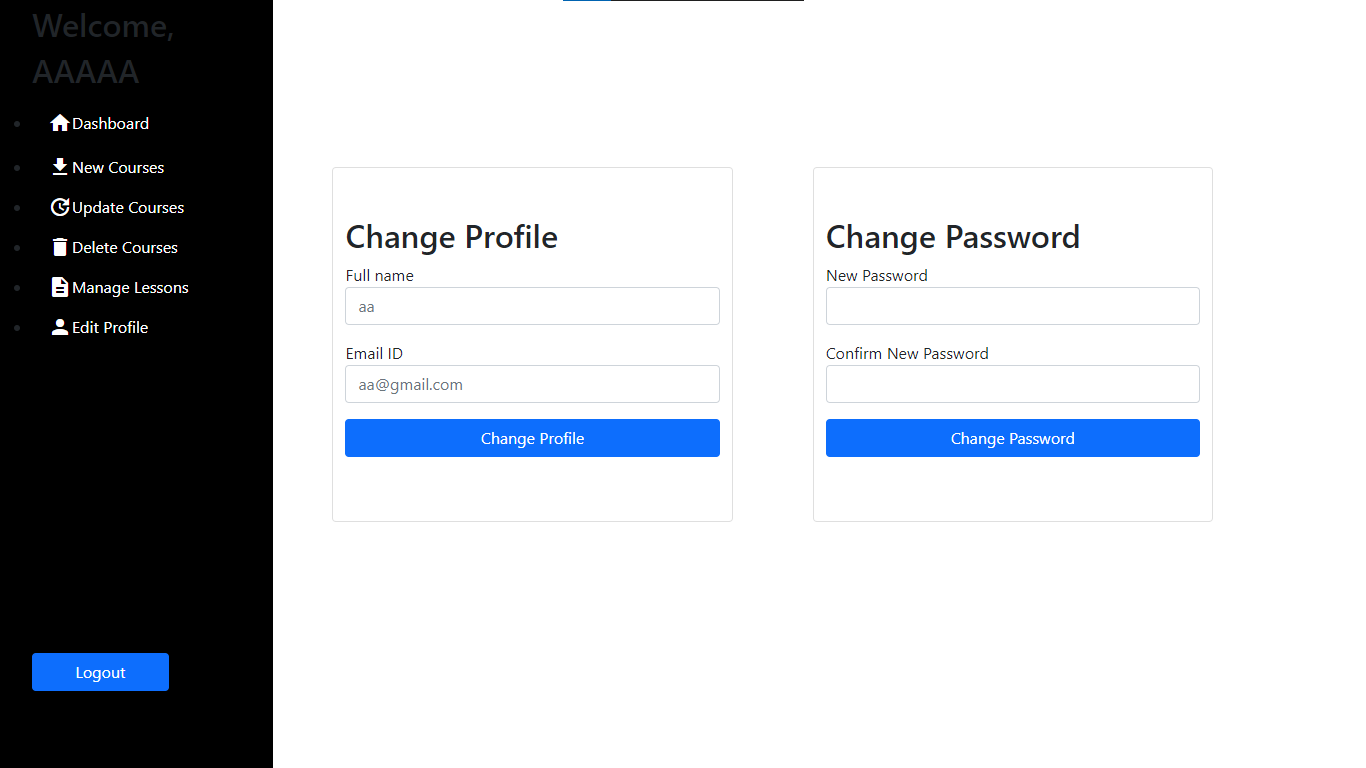
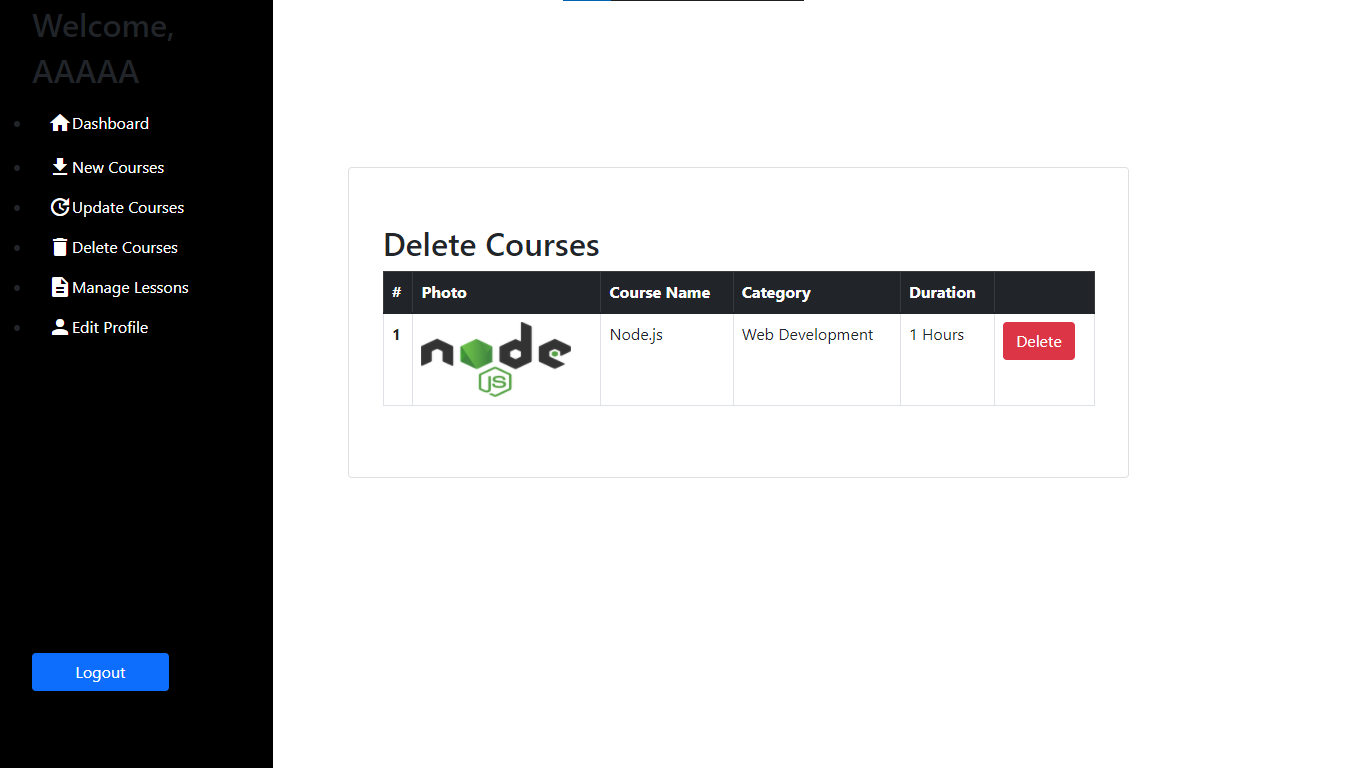
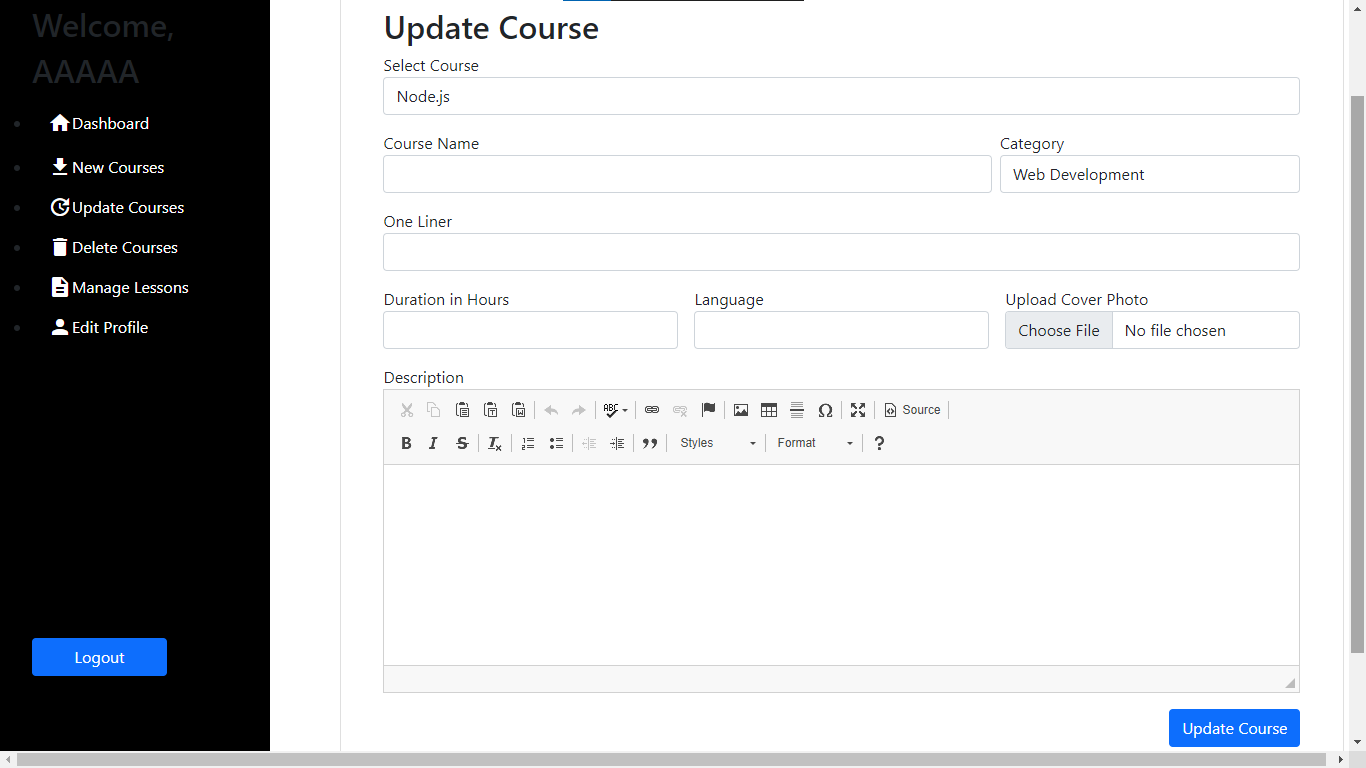
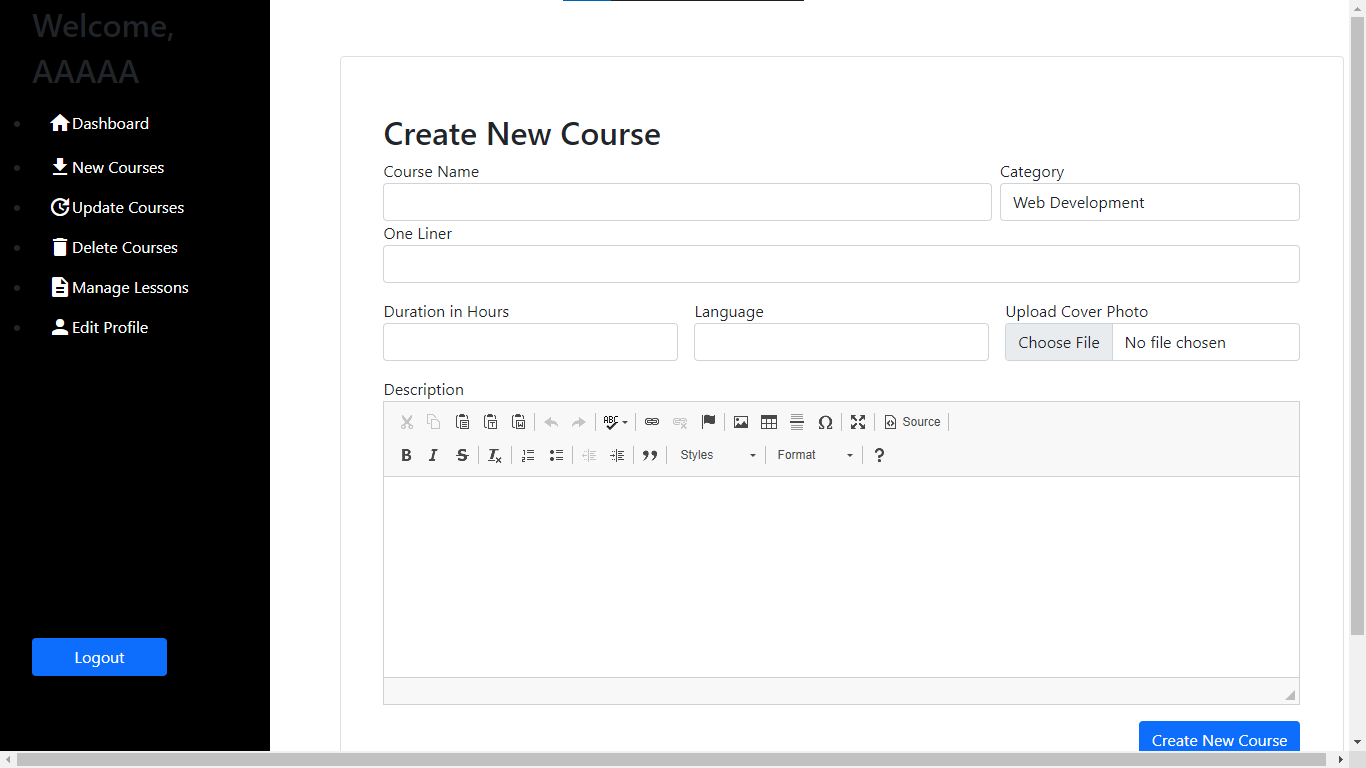
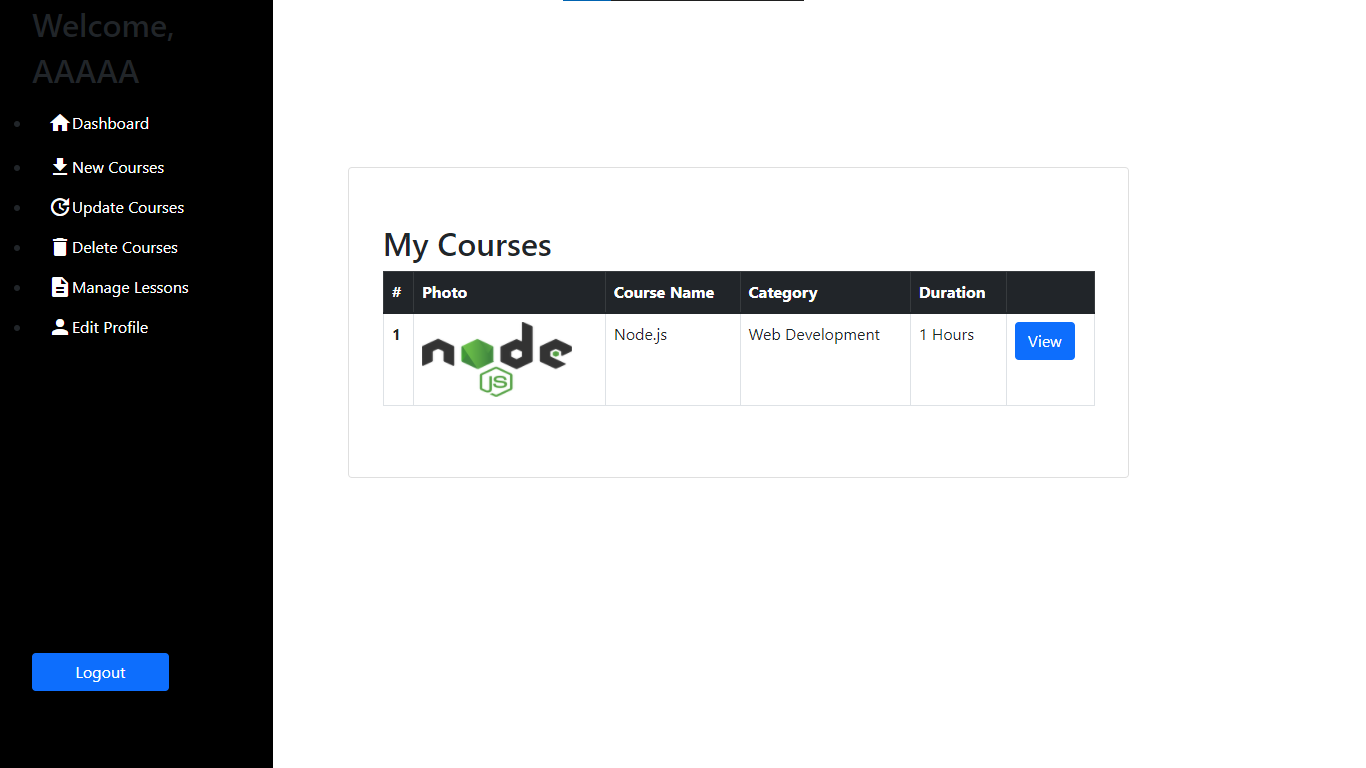
*npm i pug*

the dynamic views have to be set in the app.js file

to upload images from the form, install multer by entering the following command in terminal

*npm i multer*

|  |
| --- |
| app.js |
| …  const app = express();  app.set('view engine','pug');  app.set('views','views/dynamic');  app.use(bodyParser.urlencoded({extended: false}));  … |
| multer-config.js |
| const multer = require('multer');  exports.storage = multer.diskStorage({  destination: function (req,file,cb) {  cb(null, 'images/')  },  filename: function (req,file,cb) {  cb(null, file.originalname)  }  });  exports.upload = multer({ storage: this.storage }); |
| course\_routes.js |
| …  const courseController = require('../controllers/course\_controller');  const multerConfig = require('../configurations/multer-config');  const router = express.Router();  router.get('/new', courseController.getAddCourseForm);  router.post('/new', multerConfig.upload.single('courseCoverPhoto'), courseController.addCourse);  router.get('/dashboard', courseController.getListCourseView);  router.get('/update', courseController.getUpdateCourseView);  router.post('/update', multerConfig.upload.single('courseCoverPhoto'), courseController.updateCourse);  router.get('/delete', courseController.deleteCourse);  … |
| user\_routes.js |
| …  const userController = require('../controllers/user\_controller');  const router = express.Router();  router.get('/new', userController.getAddUserForm);  router.post('/new', userController.addUser);  router.get('/edit', userController.getEditUserView);  router.post('/edit', userController.editUser);  router.get('/delete', userController.deleteUser);  … |
| home.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Home  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  div.container  div.card(style="margin-top: 10%; padding: 5%;")  div(class="row g-2")  div.col  form#registerForm(method="POST", action="/user/new")  div.col  h2 New to EGyan? Sign up for free!  div.mb-3  div(class="row g-2")  div.col-8 Full name  input.form-control(name="userName", type="text", required)  div.col Type  select.form-control(name="userType", type="text", required)  option(value="Student") Student  option(value="Instructor") Instructor  div.mb-3 Email ID  input.form-control(name="userEmail", type="email", placeholder="you@example.com", required)  div.mb-3  div(class="row g-2")  div.col Password  input.form-control(name="userPassword", type="password", required)  div.col Confirm Password  input.form-control(type="password", required)  div(class="mb-3 d-grid gap-2")  button(class="btn btn-primary registerButton", type="submit") Register Now!  strong #{title}  div.col  form#loginForm(method="POST", action="/user/new")  div.col  h2 Have an account? Log in now!  div.mb-3 Email ID  input#userEmail.form-control(name="loginEmail", type="email", required)  div.mb-3 Password  input#userPassword.form-control(name="loginPassword", type="password", required)  div(class="mb-3 d-grid gap-2")  button(class="btn btn-primary loginButton", type="submit") Login  strong #{name} |
| navbar.pug |
| link(rel="stylesheet", href="https://fonts.googleapis.com/icon?family=Material+Icons")  div(class="d-flex flex-column", style="position: fixed; top: 0; left: 0; height:100%; width: 20%; z-index: 1; background-color: black; overflow-x: hidden;")  ul(class="nav, flex-column")  div(class="navbar-brand text-white")  h2 Welcome,  h2 #{users.userName.split(" ")[0]}  div.mb-1  a(class="nav-link text-white mb-1", href="/course/dashboard") Dashboard  span.material-icons(style="float: left;") home  div.mb-1  a(class="nav-link text-white", href="/course/new") New Courses  span.material-icons(style="float: left;") download  div.mb-1  a(class="nav-link text-white", href="/course/delete") Delete Courses  span.material-icons(style="float: left;") delete  div.nav-item  a(class="nav-link text-white disabled", href="#") Manage Lessons  span.material-icons(style="float: left;") description  div.mb-1  a(class="nav-link text-white", href="/user/edit") Edit Profile  span.material-icons(style="float: left;") person  div.mb-1  a(class="nav-link text-white", href="/user/delete") Delete Profile  span.material-icons(style="float: left;") person\_remove  a(class="btn btn-primary", type="submit", href="/user/new", style="position: fixed; bottom: 10%; width: 10%;") Logout |
| user\_edit.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Edit User  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  include navbar.pug  div.container  form(method="POST", action="/user/edit")  div(class="row justify-content-around", style="margin-top: 15%; margin-left: 15%")  div(class="card col-5",style="padding-top: 5%; padding-bottom: 5%;")  h2 Change Profile  div.mb-3 Full name  input.form-control(type="hidden" name="\_id" value=users.\_id)  input.form-control(name="userName", type="text", placeholder=users.userName, required)  div.mb-3 Email ID  input.form-control(name="userEmail", type="email", placeholder=users.userEmail, required)  div(class="mb-3 d-grid gap-2")  button(class="btn btn-primary", type="submit") Change Profile  div(class="card col-5" style="padding-top: 5%; padding-bottom: 5%;")  h2 Change Password  div.mb-3 New Password  input.form-control(name="userPassword", type="password", required)  div.mb-3 Confirm New Password  input.form-control(type="password", required)  div(class="mb-3 d-grid gap-2")  button(class="btn btn-primary", type="submit") Change Password |
| user\_delete.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title User Delete  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  include navbar.pug  h1(style="text-align: center; margin-top: 2%;") Delete Users  div.container(style="margin-top: 5%; margin-left: 20%")  table(class="table table-bordered")  thead.table-dark  tr  th(scope="col") #  th(scope="col") Name  th(scope="col") Type  th(scope="col") Email  th(scope="col") Password  th(scope="col")  tbody  each eachUser, index in userList  td=index+1  td=eachUser.userName  td=eachUser.userType  td=eachUser.userEmail  td=eachUser.userPassword  td  a(class="btn btn-danger", type="button", href="/user/delete?id="+eachUser.\_id) Delete  tr |
| course\_new.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title New Course  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  script(src="//cdn.ckeditor.com/4.17.2/standard/ckeditor.js")  body  include navbar.pug  div.container  form(method="POST", action="/course/new", enctype="multipart/form-data")  div.card(style="left: 15%; margin: 5%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 Create New Course  div.mb3  div(class="row g-2")  div.col-8 Course Name  input.form-control(name="courseName", type="text", required)  div.col Category  select.form-control(name="courseCategory")  each courseCategory in courseCategories  option=courseCategory  div.mb-3 One Liner  input.form-control(name="courseOneLiner", type="text", required)  div.mb-3  div(class="row g-3")  div.col Duration in Hours  input.form-control(name="courseDuration", type="text", required)  div.col Language  input.form-control(name="courseLanguage", type="text", required)  div#coverPhoto.col Upload Cover Photo  input.form-control(name="courseCoverPhoto", type="file", required)  div.mb-3 Description  textarea.form-control(name="courseDescription", required)  script(type="text/javascript") CKEDITOR.replace( 'courseDescription' );  button(class="btn btn-primary", type="submit", style="float: right;") Create New Course |
| dashboard.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Dashboard  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  include navbar.pug  div.container  div.card(style="left: 5%; margin: 15%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 My Courses  table(class="table table-bordered")  thead.table-dark  tr  th(scope="col") #  th(scope="col") Photo  th(scope="col") Course Name  th(scope="col") Category  th(scope="col") Duration  th(scope="col")  tbody  each course,index in courses  td=index+1  td  img(src="data:image/jpg;base64,"+course.courseCoverPhoto, style="height: 100px; width: 200px")  td=course.courseName  td=course.courseCategory  td=course.courseDuration  td  a(class="btn btn-primary", type="button", href="/course/update?id="+course.\_id) View/Update  tr |
| course\_update.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Update Course  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  script(src="//cdn.ckeditor.com/4.17.2/standard/ckeditor.js")  body  include navbar.pug  div.container  form(method="POST", action="/course/update", enctype="multipart/form-data")  div.card(style="left: 15%; margin: 5%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 Update Course  div.mb-3  div(class="row g-2")  div.col-8 Course Name  input.form-control(name="\_id", type="hidden", value=courses.\_id)  input.form-control(name="courseName", type="text", value=courses.courseName, required)  div.col Category  select.form-control(name="courseCategory", value=courses.courseCategory, required)  each courseCategory in courseCategories  option=courseCategory  div.mb-3 One Liner  input.form-control(name="courseOneLiner", type="text", value=courses.courseOneLiner, required)  div.mb-3  div(class="row g-3")  div.col Duration in Hours  input.form-control(name="courseDuration", type="text", value=courses.courseDuration, required)  div.col Language  input.form-control(name="courseLanguage", type="text", value=courses.courseLanguage, required)  div.col Upload Cover Photo  input.form-control(name="courseCoverPhoto", type="file")  label(for="courseCoverPhoto")  img(src="data:image/jpg;base64,"+courses.courseCoverPhoto, style="height: 50px; width: 100px")  div.mb-3 Description  textarea.form-control(name="courseDescription", value=courses.courseDescription, required) #{courses.courseDescription}  script(type="text/javascript") CKEDITOR.replace( 'courseDescription' );  button(class="btn btn-primary", type="submit", style="float: right;") Update Course |
| course\_delete.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Delete Course  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  include navbar.pug  div.container  div.card(style="left: 5%; margin: 15%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 Delete Courses  table(class="table table-bordered")  thead.table-dark  tr  th(scope="col") #  th(scope="col") Photo  th(scope="col") Course Name  th(scope="col") Category  th(scope="col") Duration  th(scope="col")  tbod  each course,index in courses  td=index+1  td  img(src="data:image/jpg;base64,"+course.courseCoverPhoto, style="height: 100px; width: 200px")  td=course.courseName  td=course.courseCategory  td=course.courseDuration  td  a(class="btn btn-danger", type="button", href="/course/delete?id="+course.\_id) Delete  tr |



## Create the Controllers

|  |
| --- |
| user\_controllers.js |
| const userModel = require('../models/user-model');  const userRepository = require('../repositories/user-repository');  const courseRepository = require('../repositories/course-repository');  const CurrentUserDetails=[];  exports.getAddUserForm = (req,res) => {  userRepository.getAll((users) => {  res.render('home.pug',{ users: users });  });  };  exports.addUser = (req,res) => {  if(Object.values(req.body).length===4) {  userRepository.getAll((users) => {  if(JSON.stringify(users).includes(req.body.userEmail)) {  res.render('home.pug',{ title: 'The email id is already registered! Kindly login.' });  }else{  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword);  userRepository.add(user);  userRepository.getAll((users) => {  res.render('home.pug',{ title: 'Registration successful! Kindly login to proceed.', users: users });  });  }  });  }else if(Object.values(req.body).length===2){  userRepository.getAll((users) => {  if(Object.values(users)[users.findIndex(std => std.userEmail === req.body.loginEmail)].userPassword === req.body.loginPassword) {  const id = Object.values(users)[users.findIndex(std => std.userEmail === req.body.loginEmail)].\_id.toString();  userRepository.getById(id, (result) => {  CurrentUserDetails.push(result);  global.CurrentUserDetails = CurrentUserDetails;  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ users: result, courses: courses });  });  });  }else{  res.render('home.pug',{ name: 'The login details provided are incorrect. Please register or try again.' });  }  });  }  };  exports.getEditUserView = (req,res) => {  const id = CurrentUserDetails[0].\_id.toString();  userRepository.getById(id, (result) => {  res.render('user\_edit.pug',{ users: result });  });  };  exports.editUser = (req,res) => {  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword,req.body.\_id);  userRepository.update(user,() => {  const id = req.body.\_id;  userRepository.getById(id, (result) => {  res.render('user\_edit.pug',{ users: result });  });  });  };  exports.deleteUser = (req,res) => {  if(req.query.id===undefined) {  userRepository.getAll((userList) => {  res.render('user\_delete.pug',{ userList:userList, users: CurrentUserDetails[0] });  });  }else{  const id = req.query.id;  userRepository.delete((id), () => {  userRepository.getAll((userList) => {  res.render('user\_delete.pug',{ userList:userList, users: CurrentUserDetails[0] });  });  });  }  }; |
| user-routes.js |
| const express = require('express');  const userController = require('../controllers/user\_controller');  const router = express.Router();  router.get('/new', userController.getAddUserForm);  router.post('/new', userController.addUser);  router.get('/edit', userController.getEditUserView);  router.post('/edit', userController.editUser);  router.get('/delete', userController.deleteUser);  module.exports = router; |
| course\_controllers.js |
| const fs = require('fs');  const courseModel = require('../models/course-model');  const ImageModel = require('../models/image-model');  const courseRepository = require('../repositories/course-repository');  const imageRepository = require('../repositories/image-repository');  global.courseCategories = ["Web Development", "Data Science", "Business Analysis", "Project Management", "Big Data Engineering"];  exports.getAddCourseForm = (req,res,next) => {  res.render('course\_new.pug',{ users: CurrentUserDetails[0] });  }  exports.addCourse = (req,res,next) => {  const base64 = fs.readFileSync(req.file.path,"base64");  const buffer = Buffer.from(base64, "base64");  const imageString = buffer.toString('base64');  const course = new courseModel(req.body.courseName, req.body.courseCategory, req.body.courseOneLiner, req.body.courseDuration, req.body.courseLanguage, req.body.courseDescription, req.body.courseLessons, imageString)  courseRepository.add(course);  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0], courseCategories: courseCategories });  });  }  exports.getListCourseView = (req,res,next) => {  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  }  exports.getUpdateCourseView = (req,res,next) => {  const id = req.query.id;  courseRepository.getById(id,(result) => {  res.render('course\_update.pug',{ title: 'Courses', courses: result, users: CurrentUserDetails[0], courseCategories: courseCategories });  });  }  exports.updateCourse = (req,res,next) => {  const base64 = fs.readFileSync(req.file.path,"base64");  const buffer = Buffer.from(base64, "base64");  const imageString = buffer.toString('base64');  const course = new courseModel(req.body.courseName, req.body.courseCategory, req.body.courseOneLiner, req.body.courseDuration, req.body.courseLanguage, req.body.courseDescription, req.body.courseLessons, imageString, req.body.\_id)  courseRepository.update(course,() => {  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  });  }  exports.deleteCourse = (req,res,next) => {  if(req.query.id===undefined) {  courseRepository.getAll((courses) => {  res.render('course\_delete.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  }else{  const id = req.query.id;  courseRepository.delete((id),() => {  courseRepository.getAll((courses) => {  res.render('course\_delete.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  })  }  };  exports.getImage = (req,res,next) => {  imageRepository.getAll((images) => {  res.render('course\_image.pug', { images: images });  });  };  exports.uploadImage = (req,res,next) => {  const base64 = fs.readFileSync(req.file.path,"base64");  const buffer = Buffer.from(base64, "base64");  const imageString = buffer.toString('base64');  console.log(req.file.mimetype)  const image = new ImageModel (imageString,req.file.mimetype);  imageRepository.add(image);  imageRepository.getAll((images) => {  res.render('course\_image.pug', { images: images })  });  }; |
| course-routes.js |
| const express = require('express');  const courseController = require('../controllers/course\_controller');  const multerConfig = require('../configurations/multer-config');  const router = express.Router();  router.get('/new', courseController.getAddCourseForm);  router.post('/new', multerConfig.upload.single('courseCoverPhoto'), courseController.addCourse);  router.get('/dashboard', courseController.getListCourseView);  router.get('/update', courseController.getUpdateCourseView);  router.post('/update', multerConfig.upload.single('courseCoverPhoto'), courseController.updateCourse);  router.get('/delete', courseController.deleteCourse);  module.exports = router; |

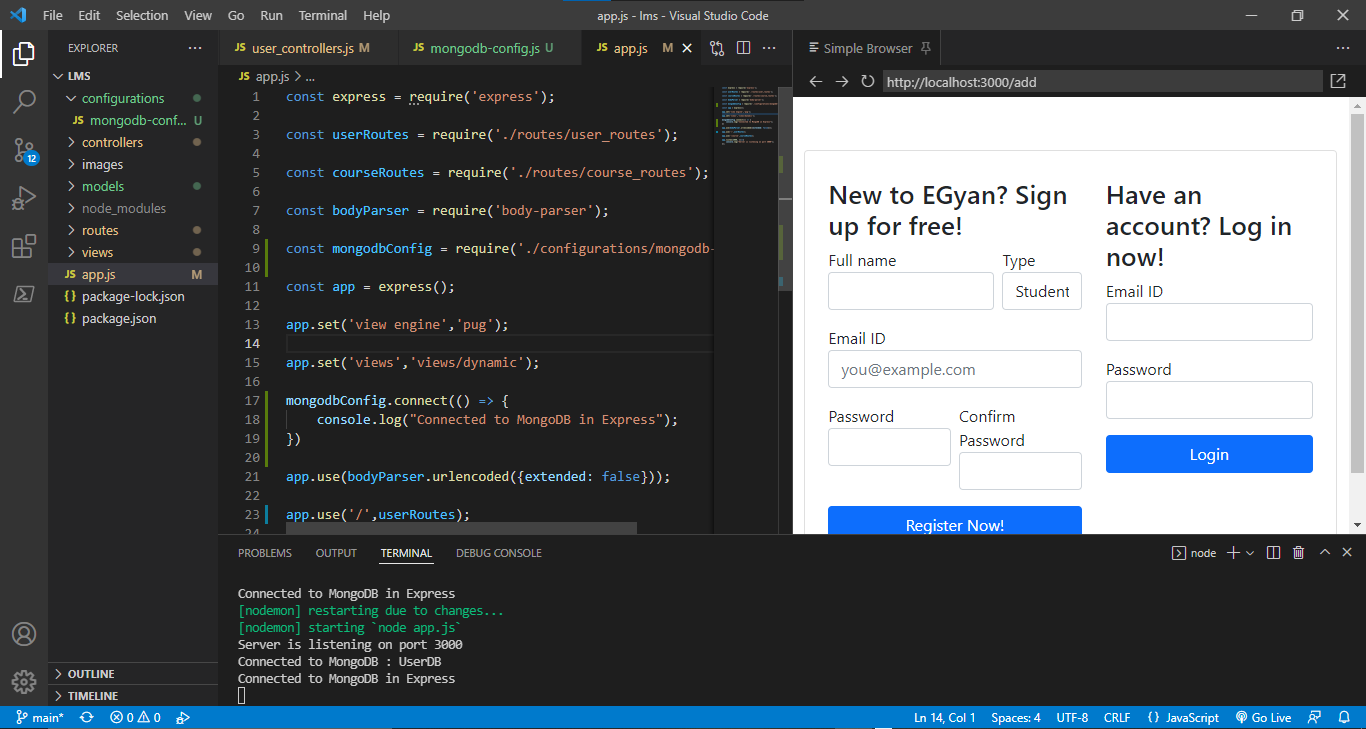
## CRUD operations for users and courses

### Create the Configuration file to connect to MongoDB

Start MongoDB in shell by typing *mongo* in terminal.

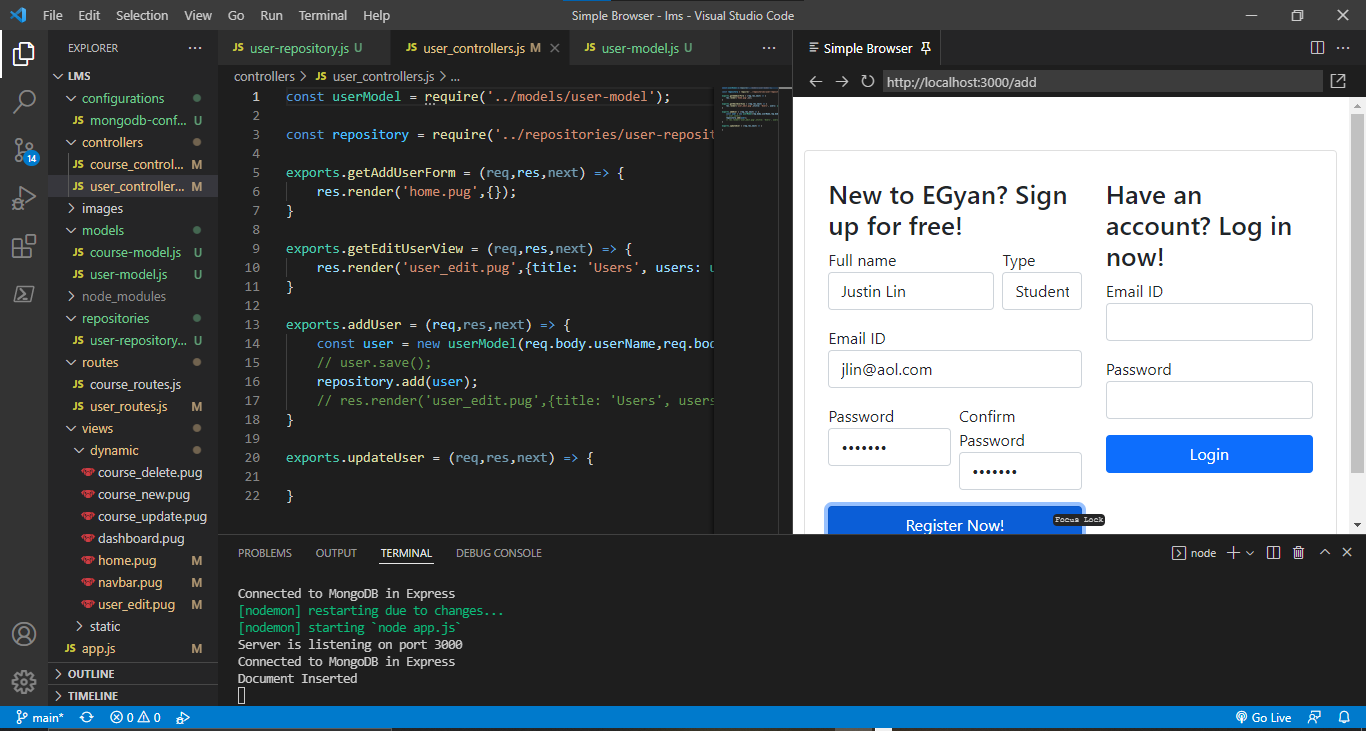
|  |
| --- |
| mongodb-config.js |
| const mongoClient = require('mongodb').MongoClient;  const uri = 'mongodb://127.0.0.1:27017/LMSDB';  const client = new mongoClient(uri);  module.exports = {  connect: function(callback) {  mongoClient.connect(uri)  .then(function(client){  console.log("Connected to MongoDB : UserDB");  return callback("OK");  })  .catch(function(err){  console.log(err);  })  }  } |
| app.js |
| …  const bodyParser = require('body-parser');  const mongodbConfig = require('./configurations/mongodb-config');  const app = express();  …  app.set('views','views/dynamic');  mongodbConfig.connect(() => {  console.log("Connected to MongoDB in Express");  })  app.use(bodyParser.urlencoded({extended: false}));  … |

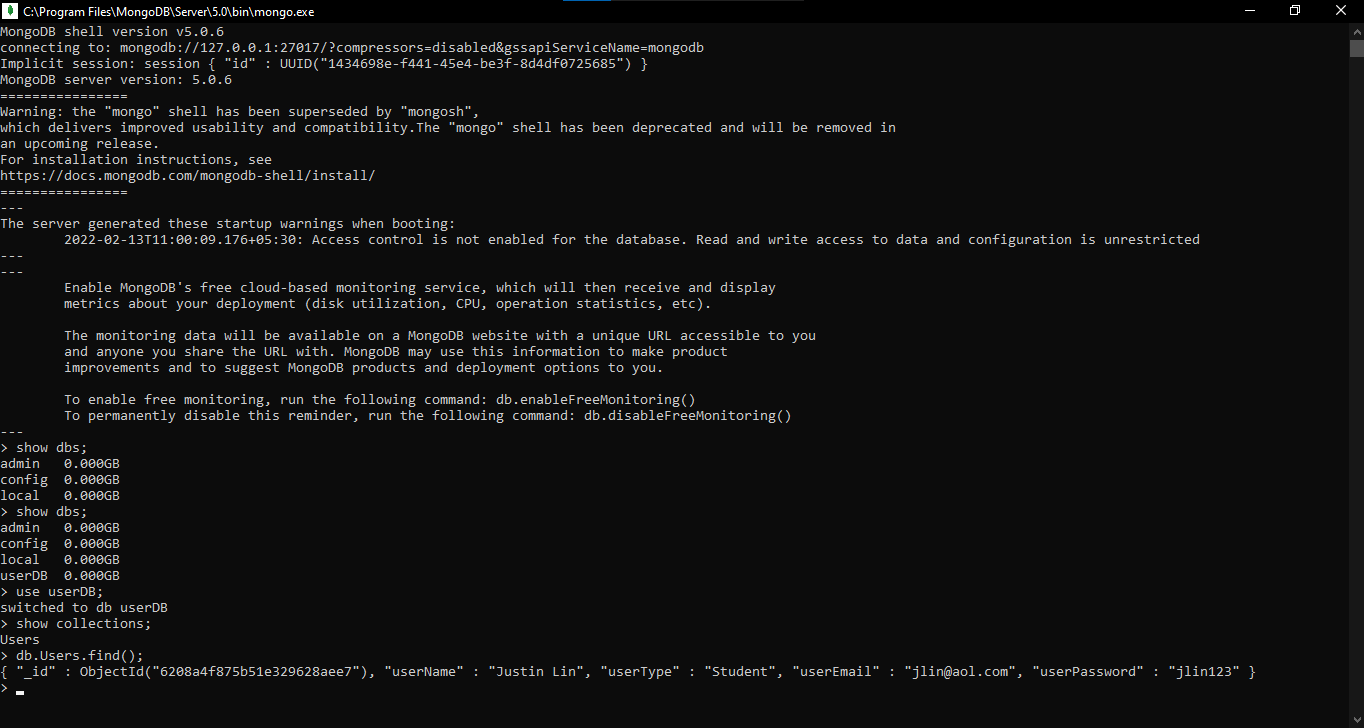
Once saved, the app will connect to the mongodb session active and return the connection message in the console (“Connected to MongoDB : UserDB­” and (“Connected to MongoDB in Express”).



### Create Operation for users

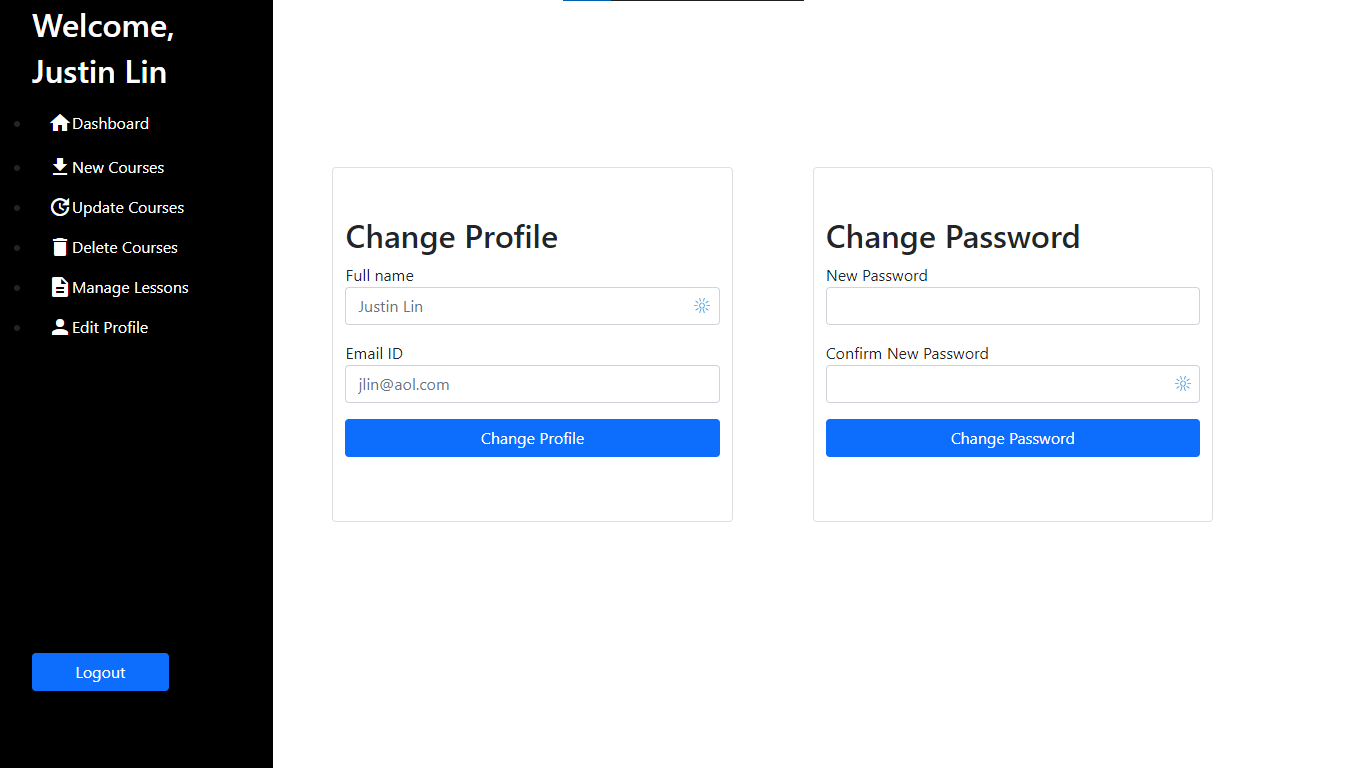
|  |
| --- |
| mongodb-config.js |
| …  const client = new mongoClient(uri);  var collection;  module.exports = {  connect: function(callback) {  mongoClient.connect(uri)  .then(function(client){  collection = client.db(‘LMSDB’).collection("Users");  return callback("OK");  })  .catch(function(err){  console.log(err);  })  },  getCollection: function(){  return collection;  }  } |
| user-repository.js |
| const database = require('../configurations/mongodb-config');  exports.add = (user,callback) => {  const collection = database.getCollection();  collection.insertOne({ userName: user.userName, userType: user.userType, userEmail: user.userEmail, userPassword: user.userPassword })  .then(() => {  console.log("Document Inserted");  })  } |
| user\_controllers.js |
| const userModel = require('../models/user-model');  const repository = require('../repositories/user-repository');  …  exports.addUser = (req,res) => {  if(Object.values(req.body).length===4) {  userRepository.getAll((users) => {  if(JSON.stringify(users).includes(req.body.userEmail)) {  res.render('home.pug',{ title: 'The email id is already registered! Kindly login.' });  }else{  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword);  userRepository.add(user);  userRepository.getAll((users) => {  res.render('home.pug',{ title: 'Registration successful! Kindly login to proceed.', users: users });  });  }  });  }else if(Object.values(req.body).length===2){  userRepository.getAll((users) => {  if(Object.values(users)[users.findIndex(std => std.userEmail === req.body.loginEmail)].userPassword === req.body.loginPassword) {  const id = Object.values(users)[users.findIndex(std => std.userEmail === req.body.loginEmail)].\_id.toString();  userRepository.getById(id, (result) => {  CurrentUserDetails.push(result);  global.CurrentUserDetails = CurrentUserDetails;  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ users: result, courses: courses });  });  });  }else{  res.render('home.pug',{ name: 'The login details provided are incorrect. Please register or try again.' });  }  });  }  };  … |





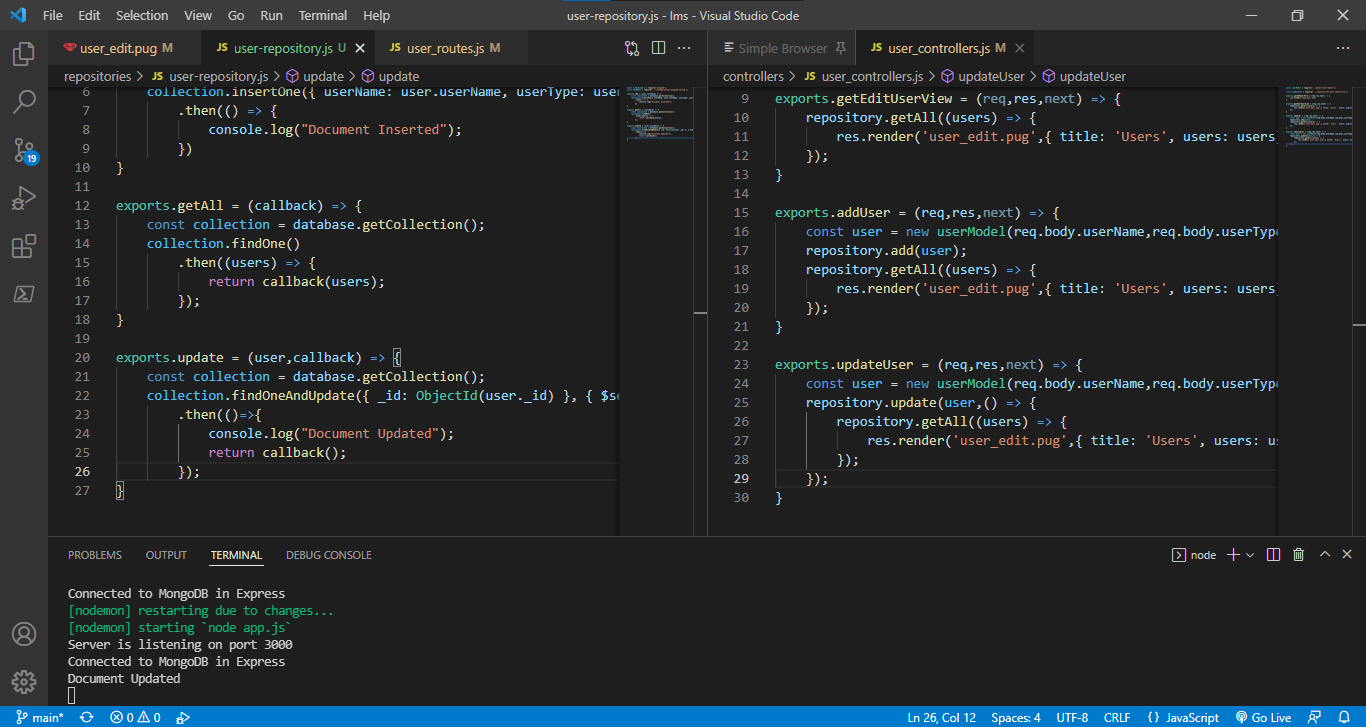
### Read Operation for users

|  |
| --- |
| user-repository.js |
| …  exports.getAll = (callback) => {  const collection = database.getCollection();  collection.findOne()  .then((users) => {  return callback(users);  });  } |
| user\_controllers.js |
| …  exports.getEditUserView = (req,res,next) => {  repository.getAll((users) => {  res.render('user\_edit.pug',{ title: 'Users', users: users});  });  }  exports.addUser = (req,res,next) => {  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword)  repository.add(user);  repository.getAll((users) => {  res.render('user\_edit.pug',{ title: 'Users', users: users});  });  }  … |
| navbar.pug |
| div(class="d-flex flex-column", style="position: fixed; top: 0; left: 0; height:100%; width: 20%; z-index: 1; background-color: black; overflow-x: hidden;")  ul(class="nav, flex-column")  div(class="navbar-brand text-white")  h2 Welcome,  h2 #{users.userName}  … |
| edit-user.pug |
| …  h2 Change Profile  div.mb-3 Full name  input.form-control(name="userName", type="text", placeholder=users.userName)  div.mb-3 Email ID  input.form-control(name="userPassword", type="password", placeholder=users.userEmail) |



### Update Operation for users

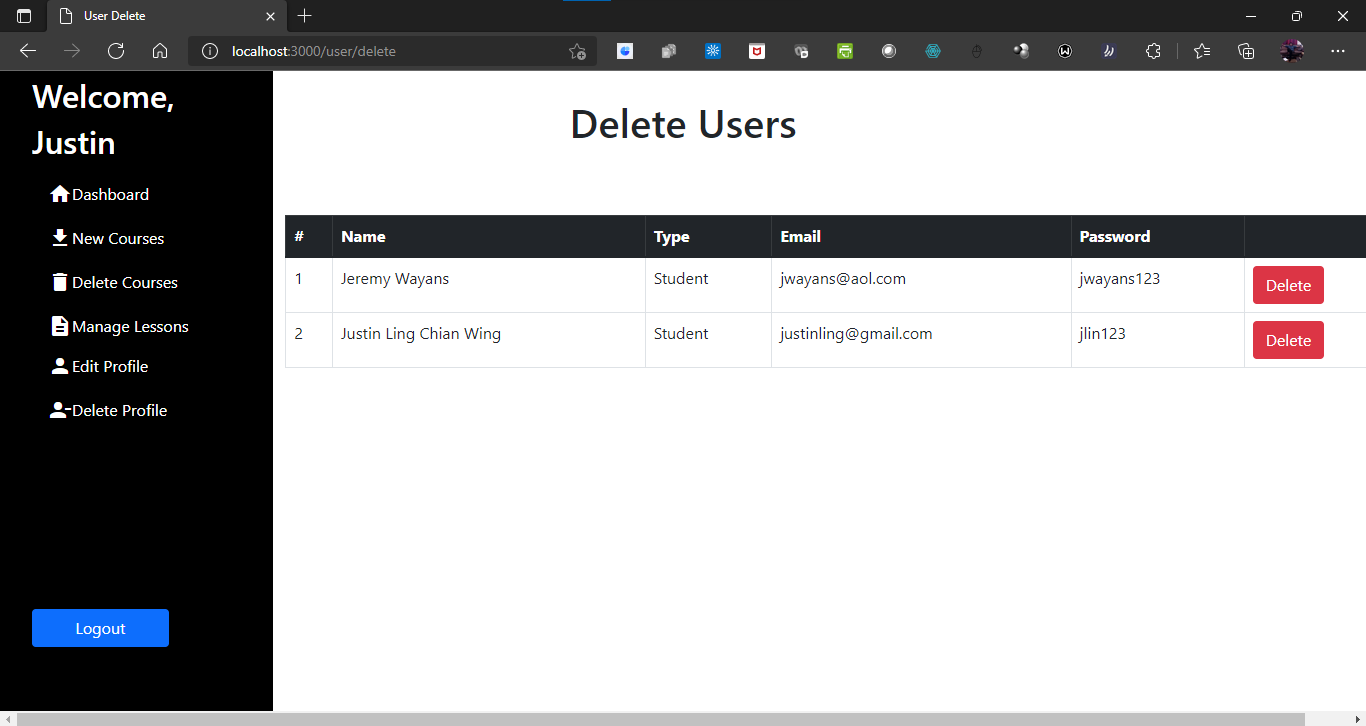
|  |
| --- |
| user-model.js |
| const users = [];  module.exports = class userModels{    constructor(userName,userType,userEmail,userPassword,id){  this.userName = userName;  this.userType = userType;  this.userEmail = userEmail;  this.userPassword = userPassword;  this.\_id = id;  }  } |
| user-routes.js |
| …  router.get('/edit', userController.getEditUserView);  router.post('/edit', userController.updateUser);  module.exports = router; |
| user-controller.js |
| …  exports.updateUser = (req,res,next) => {  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword,req.body.\_id);  repository.update(user,() => {  repository.getAll((users) => {  res.render('user\_edit.pug',{ title: 'Users', users: users});  });  });  } |
| user-repository.js |
| …  exports.update = (user,callback) => {  const collection = database.getCollection();  collection.findOneAndUpdate({ \_id: ObjectId(user.\_id) },  { $set: { userName: user.userName, userEmail: user.userEmail, userPassword: user.userPassword }},  {})  .then(()=>{  console.log("Document Updated");  return callback();  });  } |





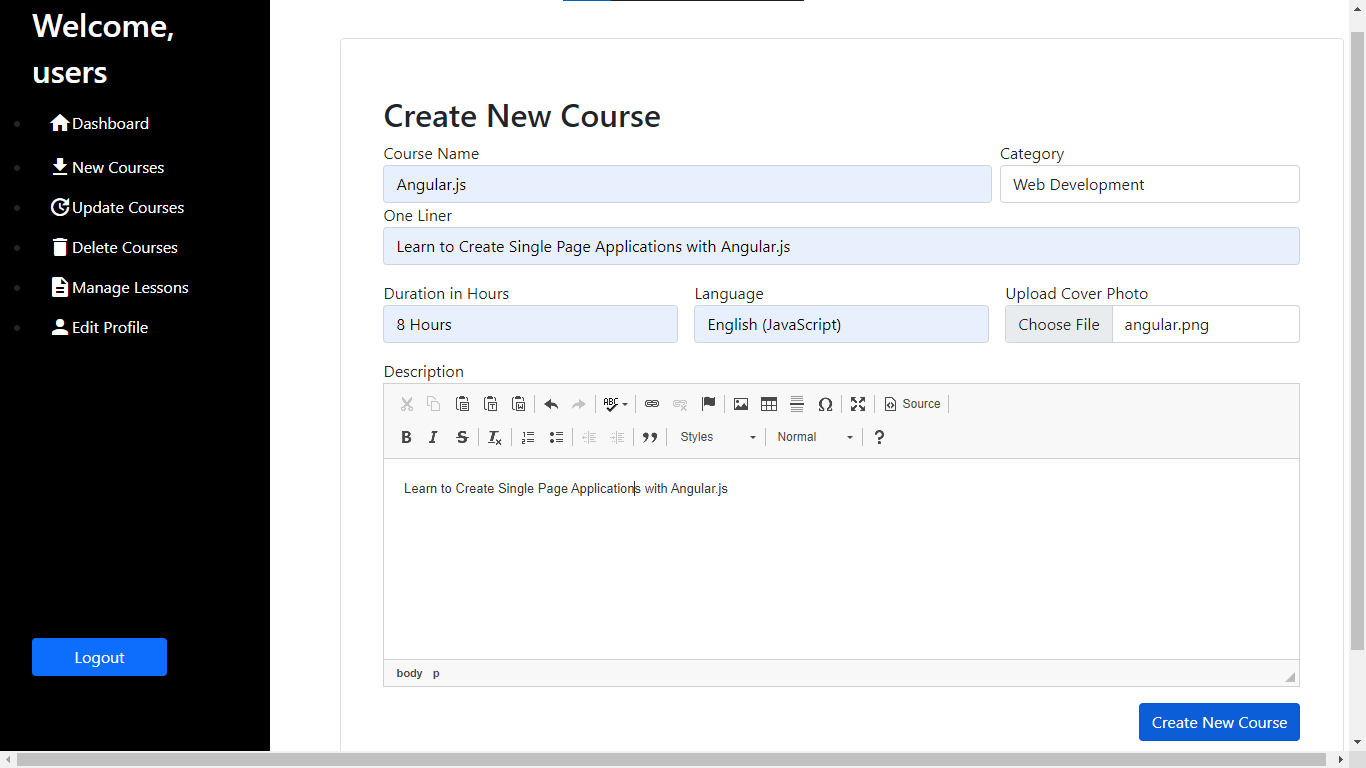
### Delete Operation for users

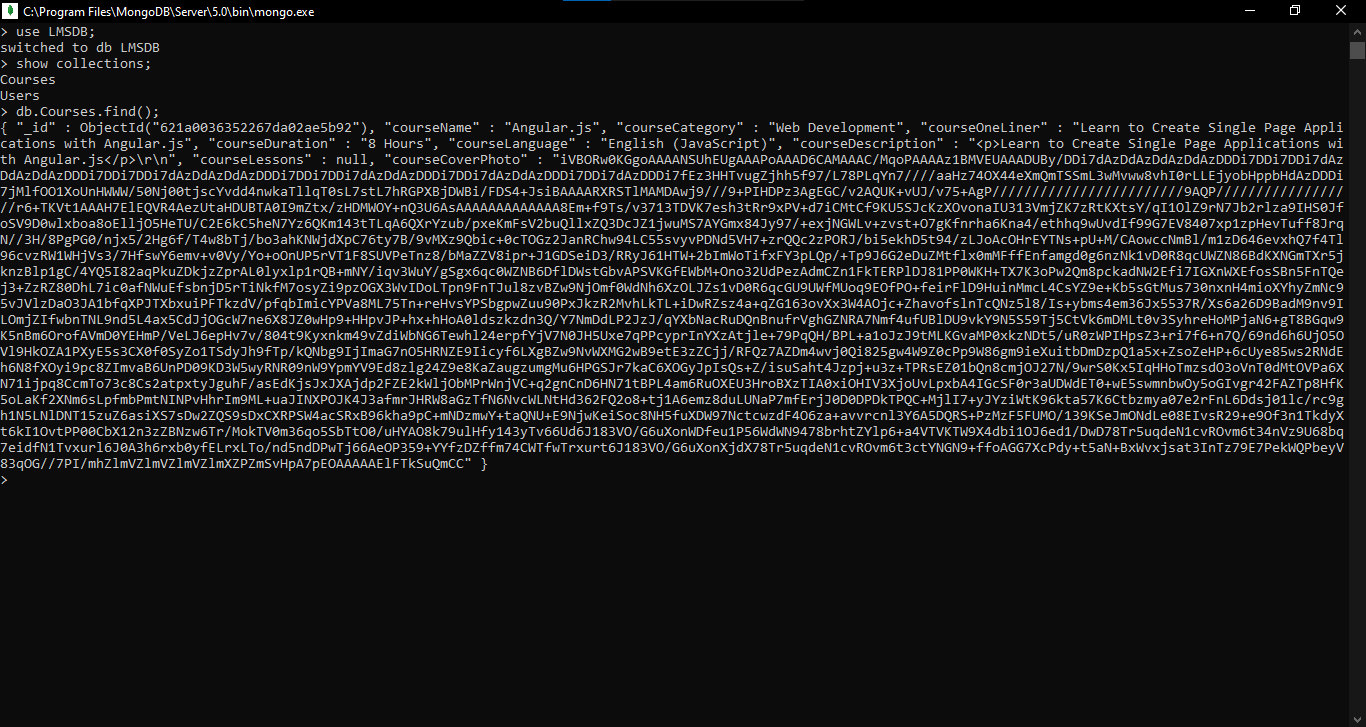
|  |
| --- |
| user-routes.js |
| …  router.get('/delete', userController.deleteUser);  module.exports = router; |
| user-controller.js |
| …  exports.deleteUser = (req,res) => {  if(req.query.id===undefined) {  userRepository.getAll((userList) => {  res.render('user\_delete.pug',{ userList:userList, users: CurrentUserDetails[0] });  });  }else{  const id = req.query.id;  userRepository.delete((id), () => {  userRepository.getAll((userList) => {  res.render('user\_delete.pug',{ userList:userList, users: CurrentUserDetails[0] });  });  });  }  }; |
| user-repository.js |
| …  exports.delete = (id,callback) => {  const userCollection = database.getUserCollection();  userCollection.deleteOne({ \_id: ObjectId(id) })  .then((result) => {  return callback();  });  }; |



### Create operation for courses

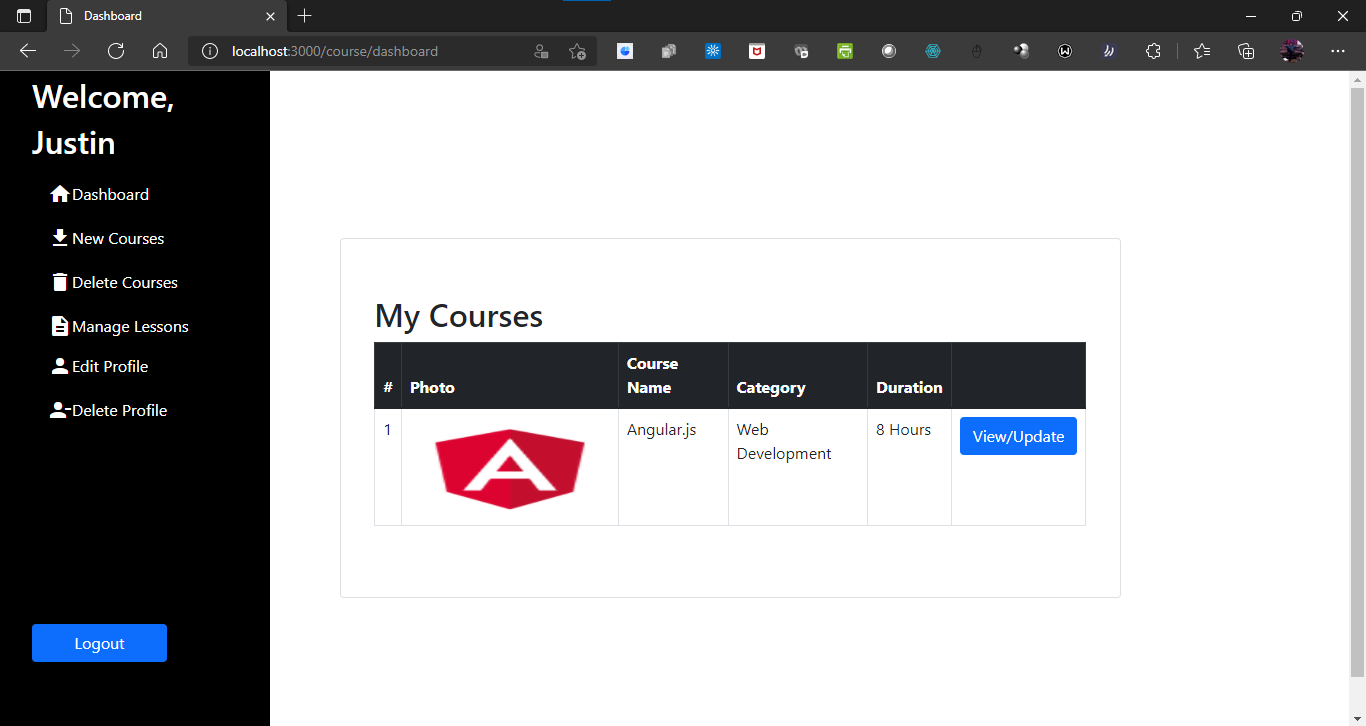
|  |
| --- |
| mongodb-config.js |
| …  module.exports = {  connect: function(callback) {  mongoClient.connect(uri)  .then(function(client){  collection = client.db(‘LMSDB’).collection("Users");  courseCollection = client.db(‘LMSDB’).collection("Courses");  return callback("OK");  })  .catch(function(err){  console.log(err);  })  },  getCollection: function(){  return collection;  },  getCourseCollection: function(){  return courseCollection;  }  } |
| course-repository.js |
| const database = require('../configurations/mongodb-config');  exports.add = (user,callback) => {  const courseCollection = database.getCourseCollection();  courseCollection.insertOne({ courseName: course.courseName, courseCategory: course.courseCategory, courseOneLiner: course.courseOneLiner, courseDuration: course.courseDuration, courseLanguage: course.courseLanguage, courseDescription: course.courseDescription, courseLessons: course.courseLessons, courseCoverPhoto: course.courseCoverPhoto })  }  exports.getAll = (user,callback) => {  const courseCollection = database.getCourseCollection();  courseCollection.find ().toArray()  .then((users) => {  return callback(courses);  });  } |
| course-contollers.js |
| const fs = require('fs');  const courseModel = require('../models/course-model');  const courseRepository = require('../repositories/course-repository');  global.courseCategories = ["Web Development", "Data Science", "Business Analysis", "Project Management", "Big Data Engineering"];  exports.getAddCourseForm = (req,res,next) => {  res.render('course\_new.pug',{ users: CurrentUserDetails[0] });  }  exports.addCourse = (req,res,next) => {  const base64 = fs.readFileSync(req.file.path,"base64");  const buffer = Buffer.from(base64, "base64");  const imageString = buffer.toString('base64');  const course = new courseModel(req.body.courseName, req.body.courseCategory, req.body.courseOneLiner, req.body.courseDuration, req.body.courseLanguage, req.body.courseDescription, req.body.courseLessons, imageString)  courseRepository.add(course);  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0], courseCategories: courseCategories });  });  } |
| course-routes.js |
| …  router.get('/new', courseController.getAddCourseForm);  router.post('/new', multerConfig.upload.single('courseCoverPhoto'), courseController.addCourse); |
| course\_new.pug |
| …  div.container  form(method="POST", action="/course/new", enctype="multipart/form-data")  div.card(style="left: 15%; margin: 5%; padding-top: 5%; padding-bottom: 5%;")  … |





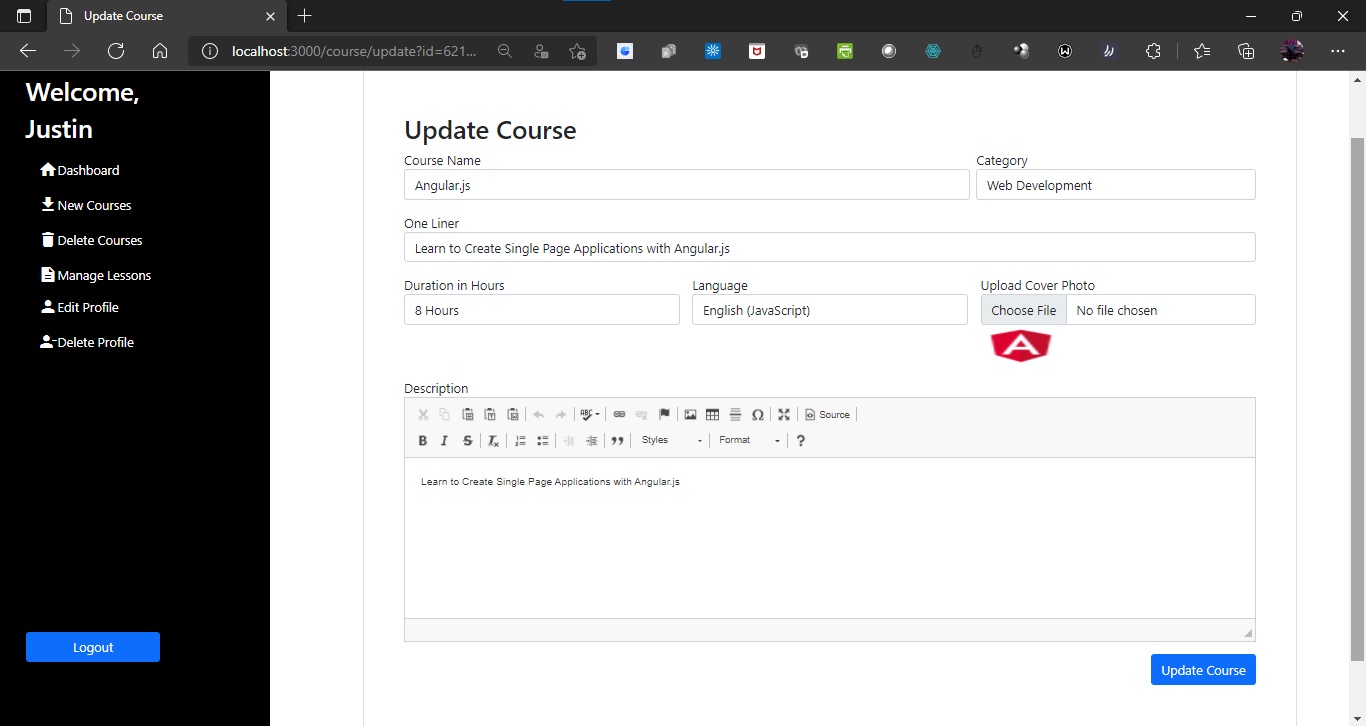
### Read operation for courses

|  |
| --- |
| course-repository.js |
| …  exports.getAll = (callback) => {  const courseCollection = database.getCourseCollection();  courseCollection.find().toArray()  .then((courses) => {  return callback(courses);  });  }; |
| course\_controller.js |
| …  exports.getListCourseView = (req,res,next) => {  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  } |
| dashboard.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Dashboard  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  include navbar.pug  div.container  div.card(style="left: 5%; margin: 15%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 My Courses  table(class="table table-bordered")  thead.table-dark  tr  th(scope="col") #  th(scope="col") Photo  th(scope="col") Course Name  th(scope="col") Category  th(scope="col") Duration  th(scope="col")  tbody  each course,index in courses  td=index+1  td  img(src="data:image/jpg;base64,"+course.courseCoverPhoto, style="height: 100px; width: 200px")  td=course.courseName  td=course.courseCategory  td=course.courseDuration  td  a(class="btn btn-primary", type="button", href="/course/update?id="+course.\_id) View/Update  tr |



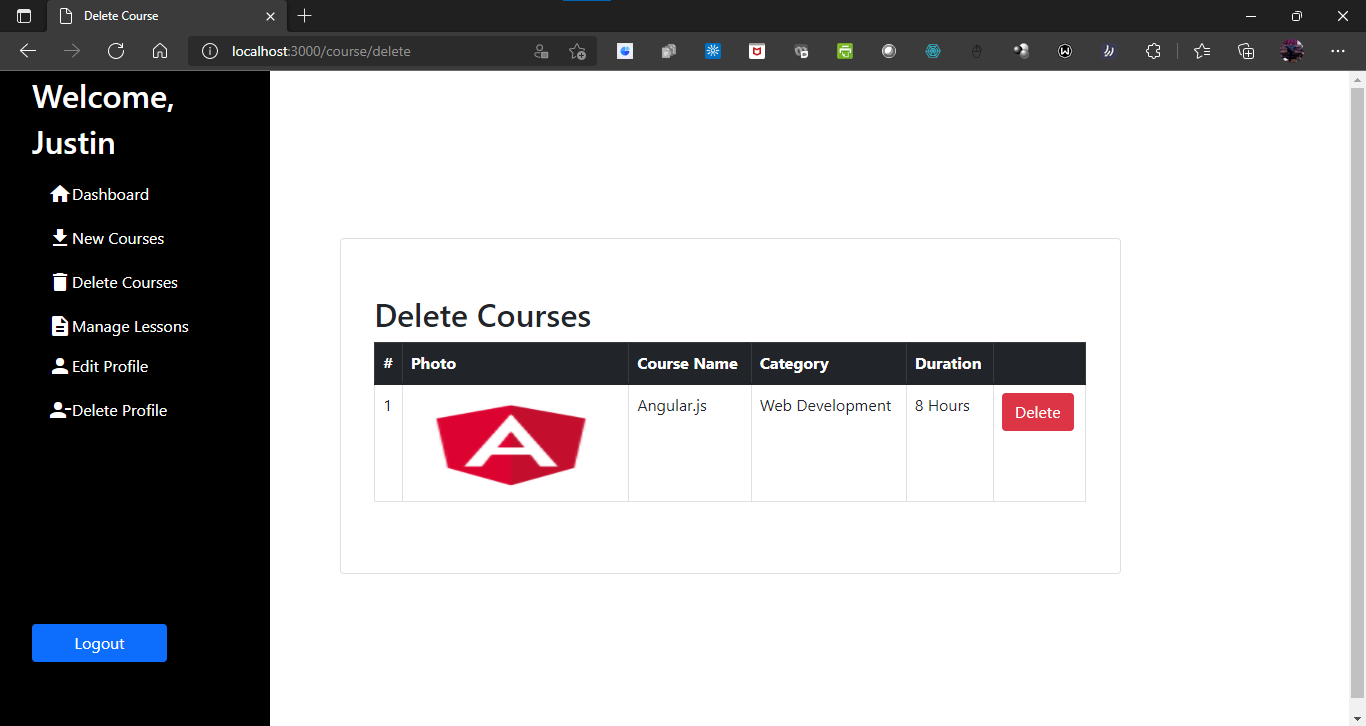
### Update operation for courses

|  |
| --- |
| course-repository.js |
| exports.getById = (id,callback) => {  const courseCollection = database.getCourseCollection();  courseCollection.findOne({ \_id: ObjectId(id) })  .then((courses) => {  return callback(courses);  });  };  exports.update = (course,callback) => {  const courseCollection = database.getCourseCollection();  courseCollection.findOneAndUpdate({ \_id: ObjectId(course.\_id) }, { $set: { courseName: course.courseName, courseCategory: course.courseCategory, courseOneLiner: course.courseOneLiner, courseDuration: course.courseDuration, courseLanguage: course.courseLanguage, courseDescription: course.courseDescription, courseLessons: course.courseLessons, courseCoverPhoto: course.courseCoverPhoto }}, {})  .then(() => {  return callback();  });  }; |
| course\_controller.js |
| …  exports.getUpdateCourseView = (req,res,next) => {  const id = req.query.id;  courseRepository.getById(id,(result) => {  res.render('course\_update.pug',{ title: 'Courses', courses: result, users: CurrentUserDetails[0], courseCategories: courseCategories });  });  }  exports.updateCourse = (req,res,next) => {  const base64 = fs.readFileSync(req.file.path,"base64");  const buffer = Buffer.from(base64, "base64");  const imageString = buffer.toString('base64');  const course = new courseModel(req.body.courseName, req.body.courseCategory, req.body.courseOneLiner, req.body.courseDuration, req.body.courseLanguage, req.body.courseDescription, req.body.courseLessons, imageString, req.body.\_id)  courseRepository.update(course,() => {  courseRepository.getAll((courses) => {  res.render('dashboard.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  });  } |
| course\_update.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Update Course  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  script(src="//cdn.ckeditor.com/4.17.2/standard/ckeditor.js")  body  include navbar.pug  div.container  form(method="POST", action="/course/update", enctype="multipart/form-data")  div.card(style="left: 15%; margin: 5%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 Update Course  div.mb-3  div(class="row g-2")  div.col-8 Course Name  input.form-control(name="\_id", type="hidden", value=courses.\_id)  input.form-control(name="courseName", type="text", value=courses.courseName, required)  div.col Category  select.form-control(name="courseCategory", value=courses.courseCategory, required)  each courseCategory in courseCategories  option=courseCategory  div.mb-3 One Liner  input.form-control(name="courseOneLiner", type="text", value=courses.courseOneLiner, required)  div.mb-3  div(class="row g-3")  div.col Duration in Hours  input.form-control(name="courseDuration", type="text", value=courses.courseDuration, required)  div.col Language  input.form-control(name="courseLanguage", type="text", value=courses.courseLanguage, required)  div.col Upload Cover Photo  input.form-control(name="courseCoverPhoto", type="file")  label(for="courseCoverPhoto")  img(src="data:image/jpg;base64,"+courses.courseCoverPhoto, style="height: 50px; width: 100px")  div.mb-3 Description  textarea.form-control(name="courseDescription", value=courses.courseDescription, required) #{courses.courseDescription}  script(type="text/javascript") CKEDITOR.replace( 'courseDescription' );  button(class="btn btn-primary", type="submit", style="float: right;") Update Course |



### Delete operation for courses

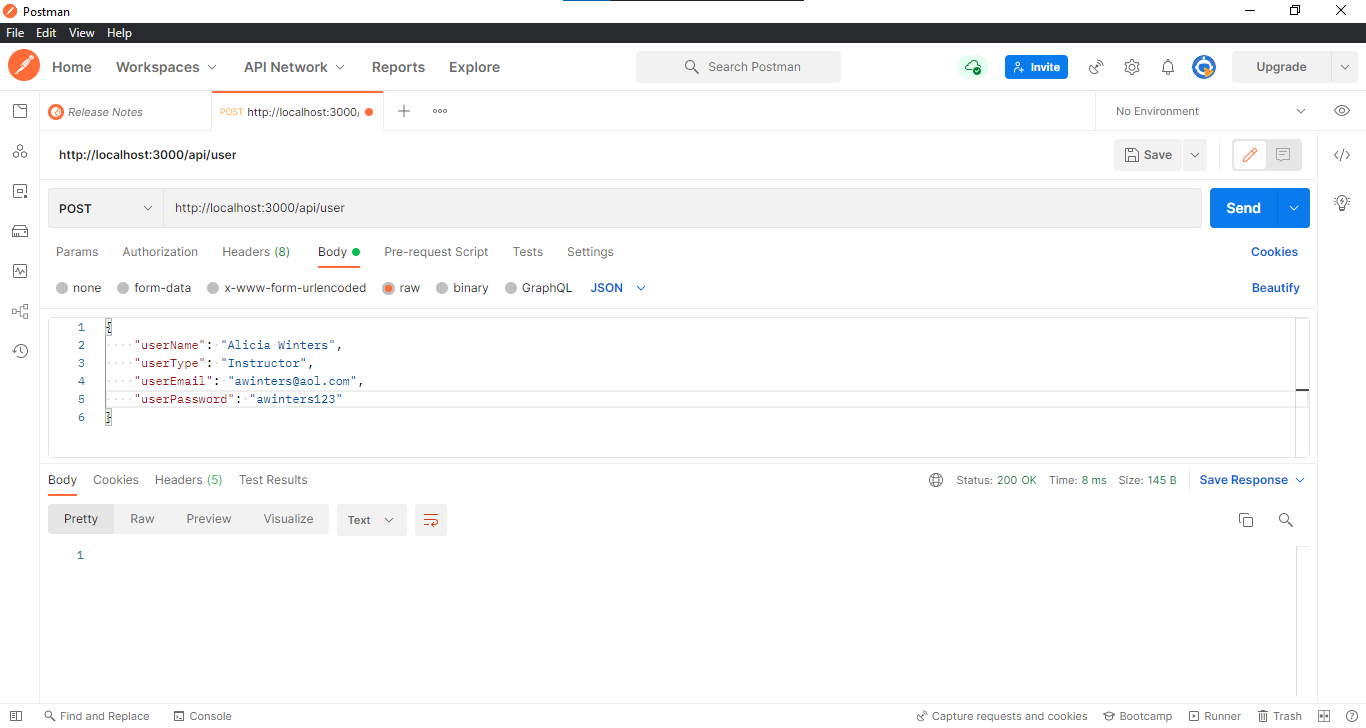
|  |
| --- |
| course-repository.js |
| exports.delete = (id,callback) => {  const courseCollection = database.getCourseCollection();  courseCollection.deleteOne({ \_id: ObjectId(id) })  .then((result) => {  return callback();  });  }; |
| course\_controller.js |
| …  exports.deleteCourse = (req,res,next) => {  if(req.query.id===undefined) {  courseRepository.getAll((courses) => {  res.render('course\_delete.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  }else{  const id = req.query.id;  courseRepository.delete((id),() => {  courseRepository.getAll((courses) => {  res.render('course\_delete.pug',{ title: 'Courses', courses: courses, users: CurrentUserDetails[0] });  });  })  }  }; |
| course\_delete.pug |
| doctype html  html(lang="en")  head  meta(charset="UTF-8")  meta(http-equiv="X-UA-Compatible", content="IE=edge")  meta(name="viewport", content="width=device-width, initial-scale=1.0")  title Delete Course  link(rel="stylesheet", href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css", crossorigin="anonymous")  body  include navbar.pug  div.container  div.card(style="left: 5%; margin: 15%; padding-top: 5%; padding-bottom: 5%;")  div(class="row justify-content-center")  div.col-11  h2 Delete Courses  table(class="table table-bordered")  thead.table-dark  tr  th(scope="col") #  th(scope="col") Photo  th(scope="col") Course Name  th(scope="col") Category  th(scope="col") Duration  th(scope="col")  tbod  each course,index in courses  td=index+1  td  img(src="data:image/jpg;base64,"+course.courseCoverPhoto, style="height: 100px; width: 200px")  td=course.courseName  td=course.courseCategory  td=course.courseDuration  td  a(class="btn btn-danger", type="button", href="/course/delete?id="+course.\_id) Delete  tr |

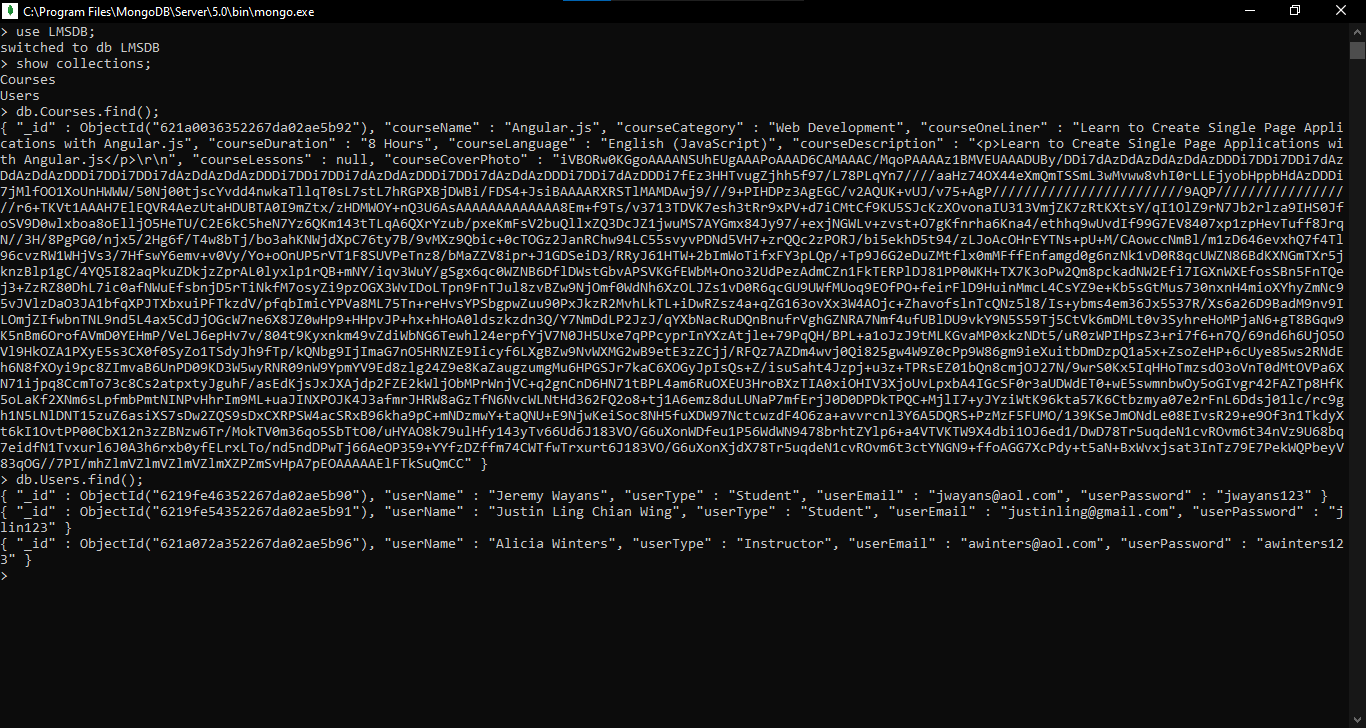


## REST API CRUD

### Create operation for users

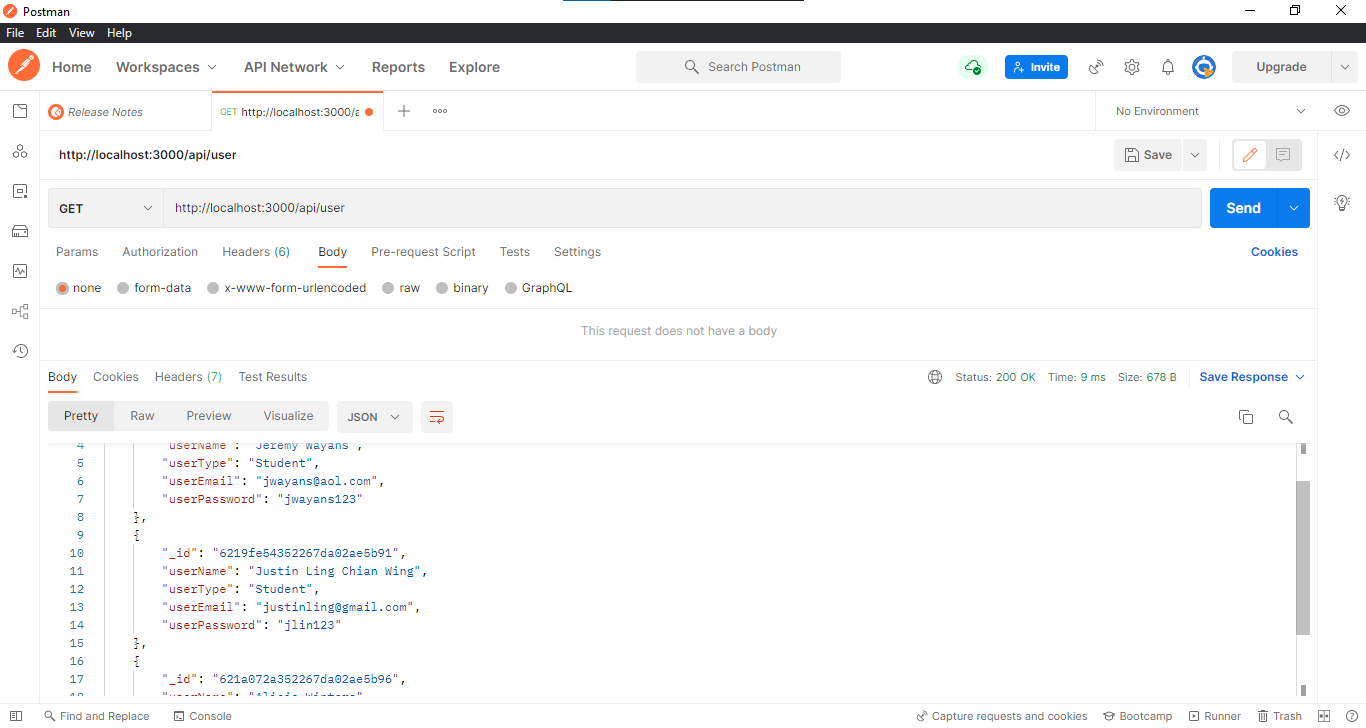
|  |
| --- |
| app.js |
| const express = require('express');  const userRoutes = require('./routes/user\_routes');  const courseRoutes = require('./routes/course\_routes');  const userRoutesAPI = require('./routes/user\_routes\_api');  const courseRoutesAPI = require('./routes/course\_routes\_api');  const bodyParser = require('body-parser');  …  app.use(bodyParser.urlencoded({extended: false}));  app.use(bodyParser.json());  …  app.use('/api/user', userRoutesAPI);  app.use('/api/course', courseRoutesAPI);  … |
| user\_routes\_api.js |
| const express = require('express');  const userControllerAPI = require('../controllers/user\_controller\_api');  const router = express.Router();  router.post("/", userControllerAPI.add);  module.exports = router; |
| user\_controller\_api.js |
| const userModel = require('../models/user-model');  const userRepository = require('../repositories/user-repository');  exports.add = (req,res) => {  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword);  userRepository.add(user);  res.status(200).send();  }; |





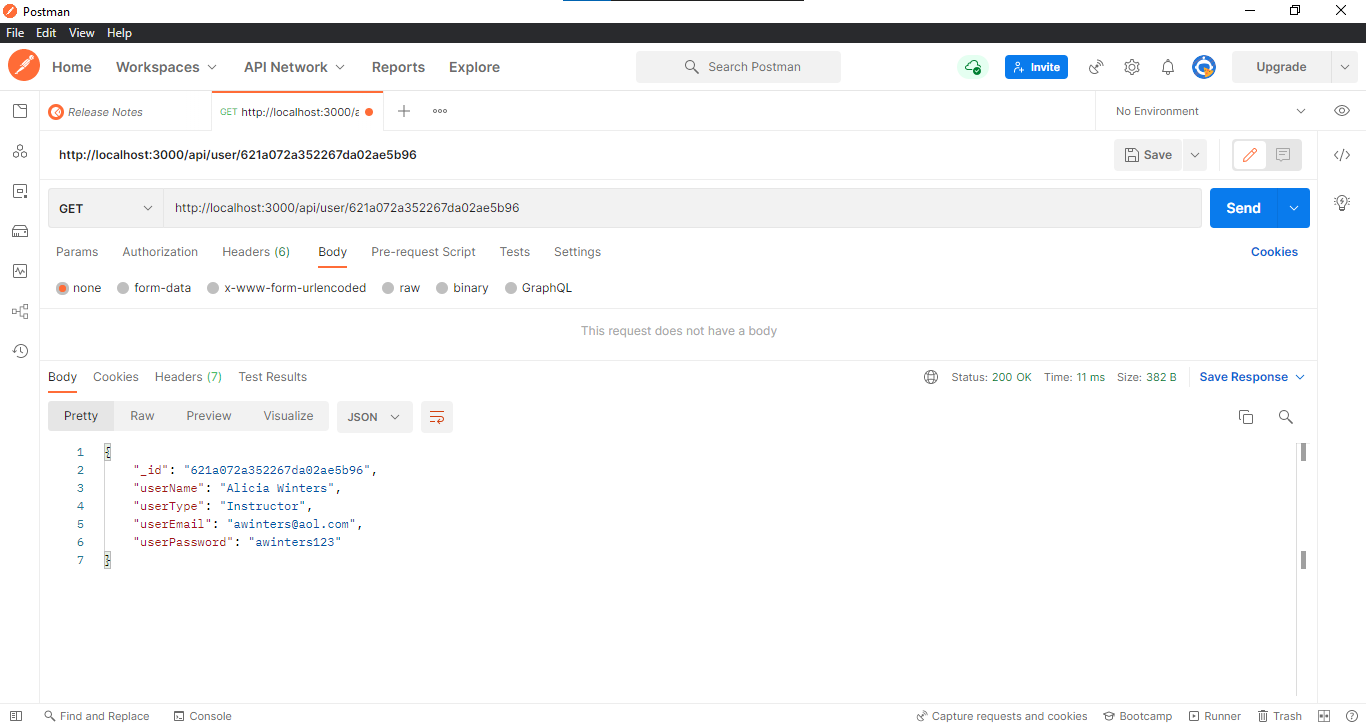
### Read operation for users

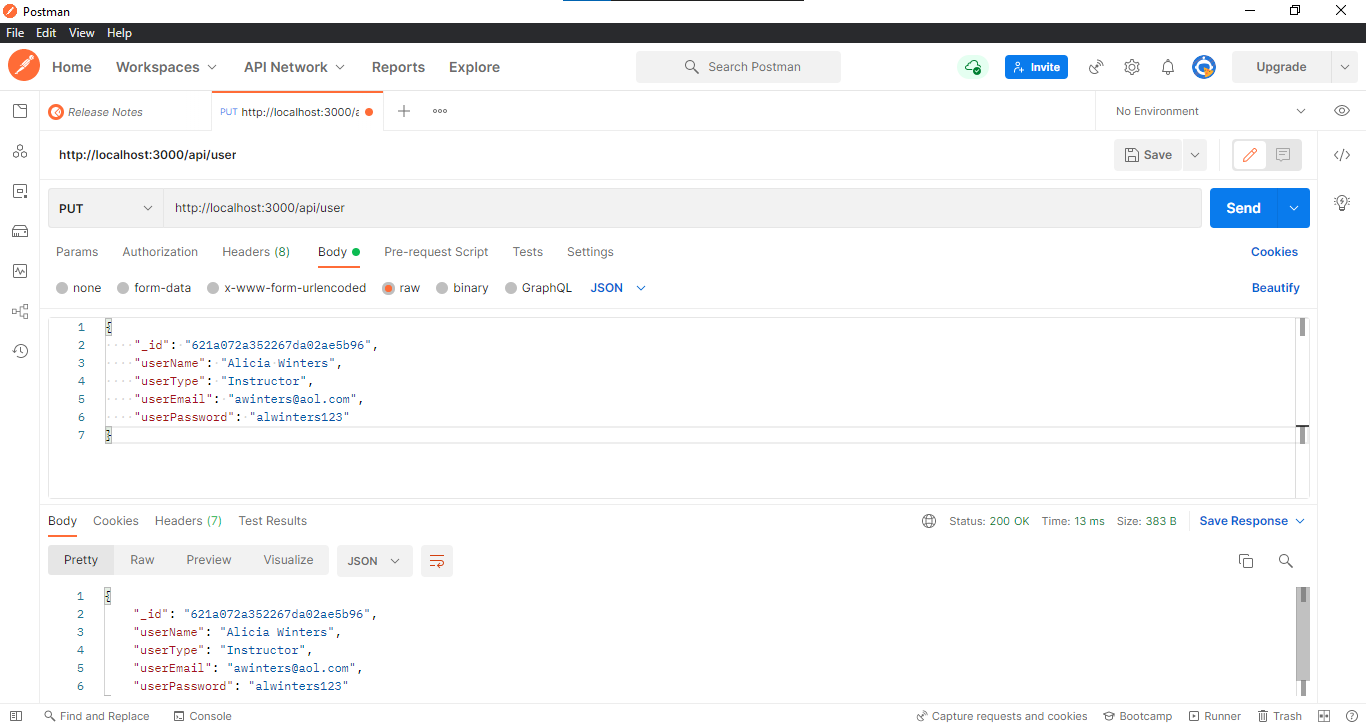
|  |
| --- |
| user\_routes\_api.js |
| …  router.get("/", userControllerAPI.getAll);  … |
| user\_controller\_api.js |
| …  exports.getAll = (req,res) => {  userRepository.getAll((users) => {  res.status(200).send(users);  });  }; |

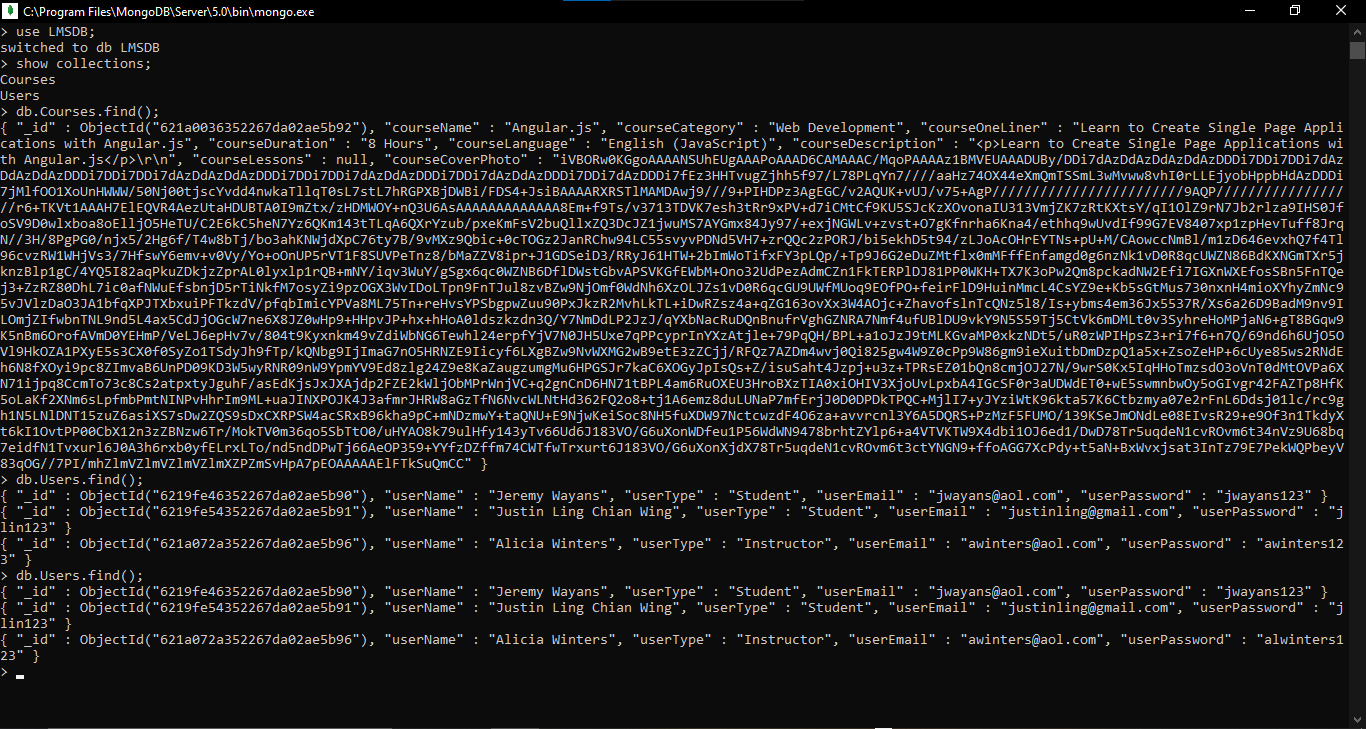


### Update operation for users

|  |
| --- |
| user\_routes\_api.js |
| …  router.get("/:id", userControllerAPI.getById);  router.put("/", userControllerAPI.update);  … |
| user\_controller\_api.js |
| …  exports.getById = (req,res) => {  const id = req.path.toString().replace("/","");  userRepository.getById(id, (user) => {  if(user){  res.status(200).send(user);  }else{  res.status(404).send("Error");  }  });  };  exports.update = (req,res) => {  const user = new userModel(req.body.userName,req.body.userType,req.body.userEmail,req.body.userPassword,req.body.\_id);  userRepository.update(user, () => {  userRepository.getById(user.\_id, (userResult) => {  res.status(200).send(userResult);  });  });  }; |

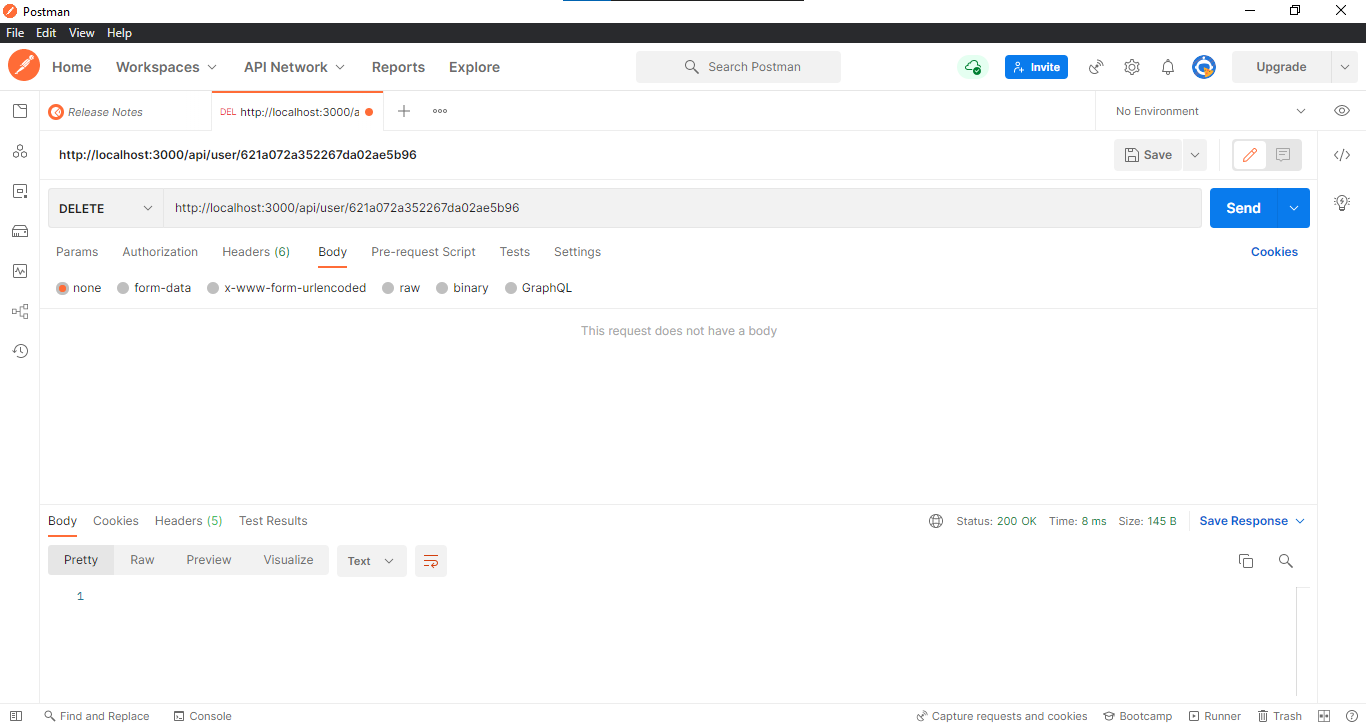


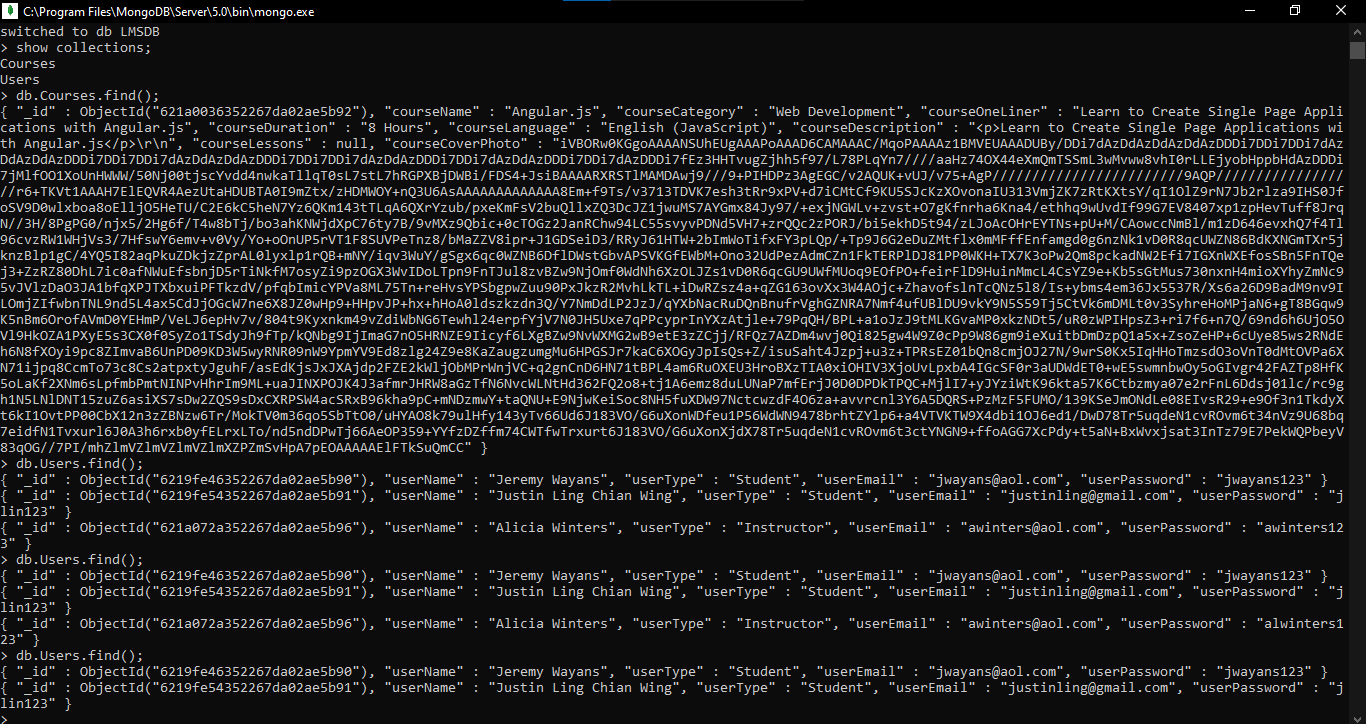




### Delete operation for users

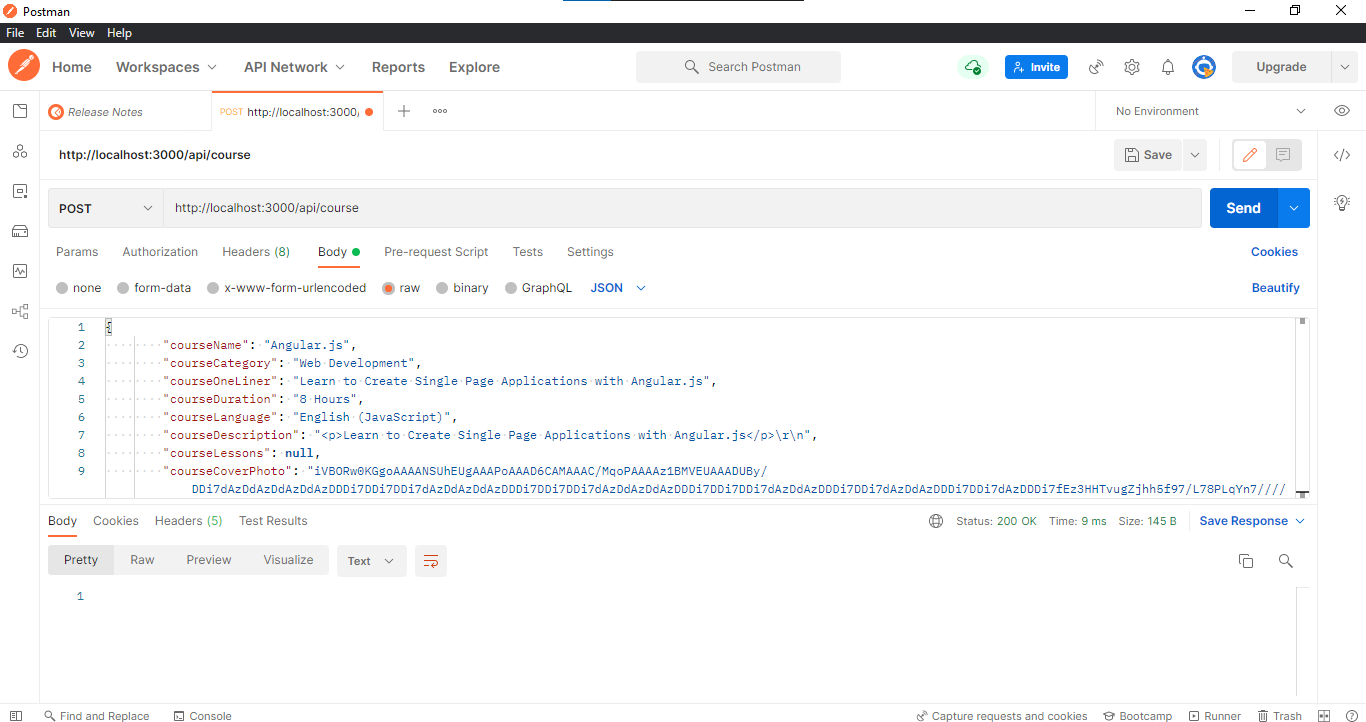
|  |
| --- |
| user\_routes\_api.js |
| …  router.delete("/:id", userControllerAPI.delete);  … |
| user\_controller\_api.js |
| …  exports.delete = (req,res) => {  const id = req.path.toString().replace("/","");  userRepository.delete(id, () => {  res.status(200).send();  });  }; |





### Create operation for courses

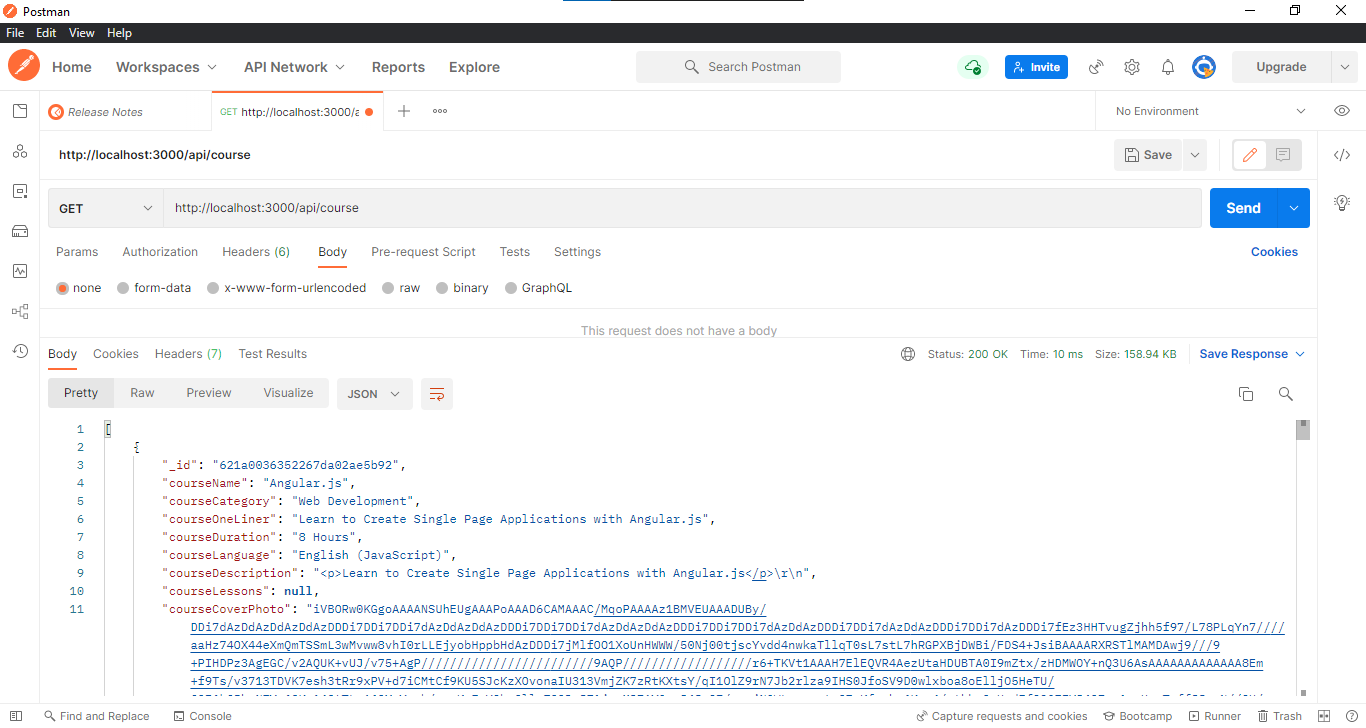
|  |
| --- |
| course\_routes\_api.js |
| const express = require('express');  const courseControllerAPI = require('../controllers/course\_controller\_api');  const router = express.Router();  router.post("/",courseControllerAPI.add);  module.exports = router; |
| course\_controller\_api.js |
| const courseRepository = require('../repositories/course-repository');  const courseModel = require('../models/course-model');  exports.add = (req,res) => {  const course = new courseModel(req.body.courseName, req.body.courseCategory, req.body.courseOneLiner, req.body.courseDuration, req.body.courseLanguage, req.body.courseDescription, req.body.courseLessons, req.body.courseCoverPhoto)  courseRepository.add(course);  res.status(200).send();  } |





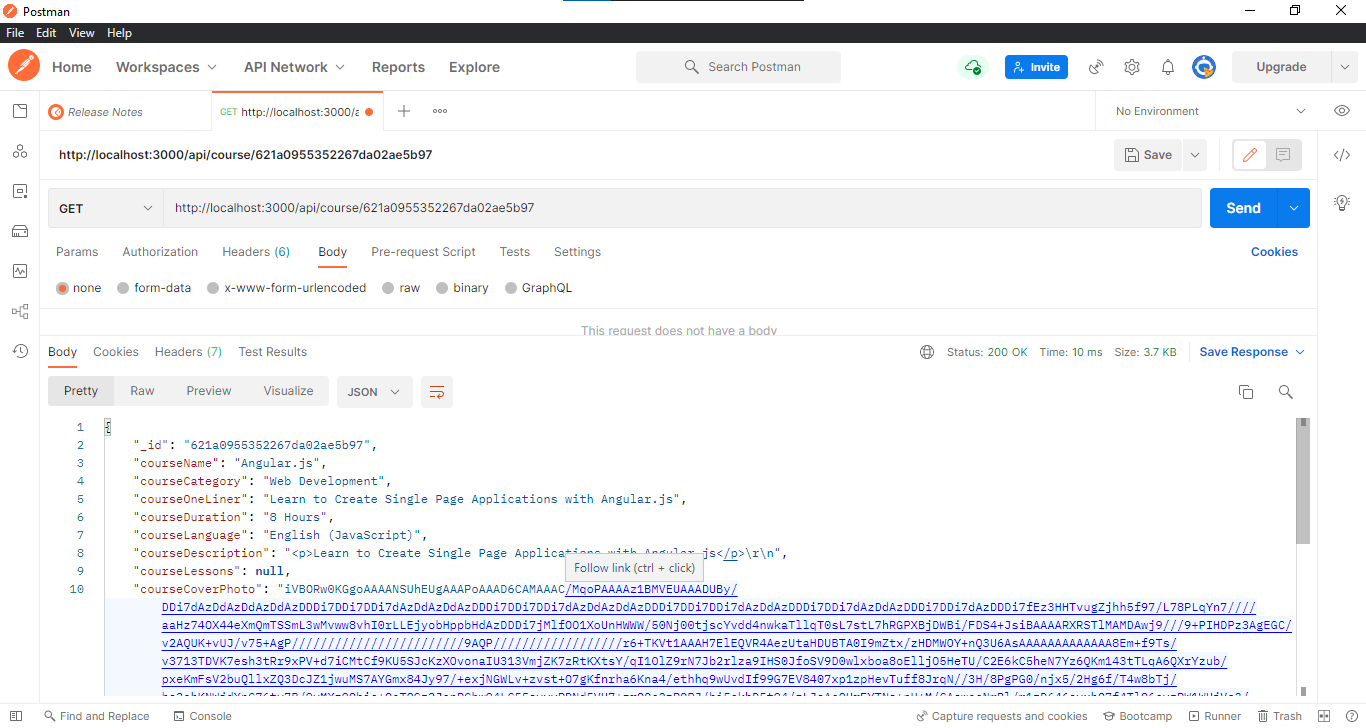
### Read operation for courses

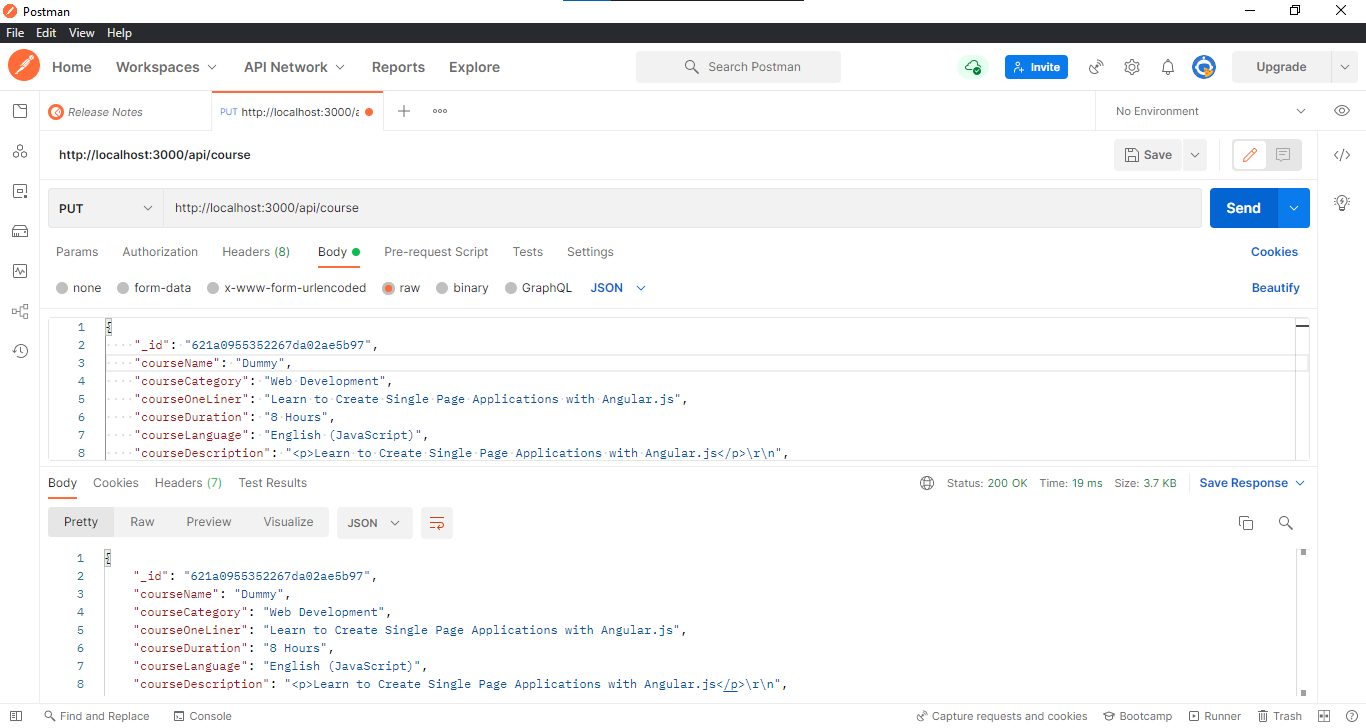
|  |
| --- |
| course\_routes\_api.js |
| …  router.get("/",courseControllerAPI.getAll);  … |
| course\_controller\_api.js |
| …  exports.getAll = (req,res) => {  courseRepository.getAll((courses) => {  res.status(200).send(courses);  });  } |



### Update operation for courses

|  |
| --- |
| course\_routes\_api.js |
| …  router.get("/:id",courseControllerAPI.getById);  router.put("/",courseControllerAPI.update);  … |
| course\_controller\_api.js |
| …  exports.getById = (req,res) => {  const id = req.path.toString().replace("/","");  courseRepository.getById(id, (course) => {  if(course){  res.status(200).send(course);  }else{  res.status(400).send("Error");  }  });  }  exports.update = (req,res) => {  const course = new courseModel(req.body.courseName, req.body.courseCategory, req.body.courseOneLiner, req.body.courseDuration, req.body.courseLanguage, req.body.courseDescription, req.body.courseLessons, req.body.courseCoverPhoto, req.body.\_id)  courseRepository.update((course), () => {  courseRepository.getById(course.\_id, (courseResult) => {  res.status(200).send(courseResult);  });  });  } |

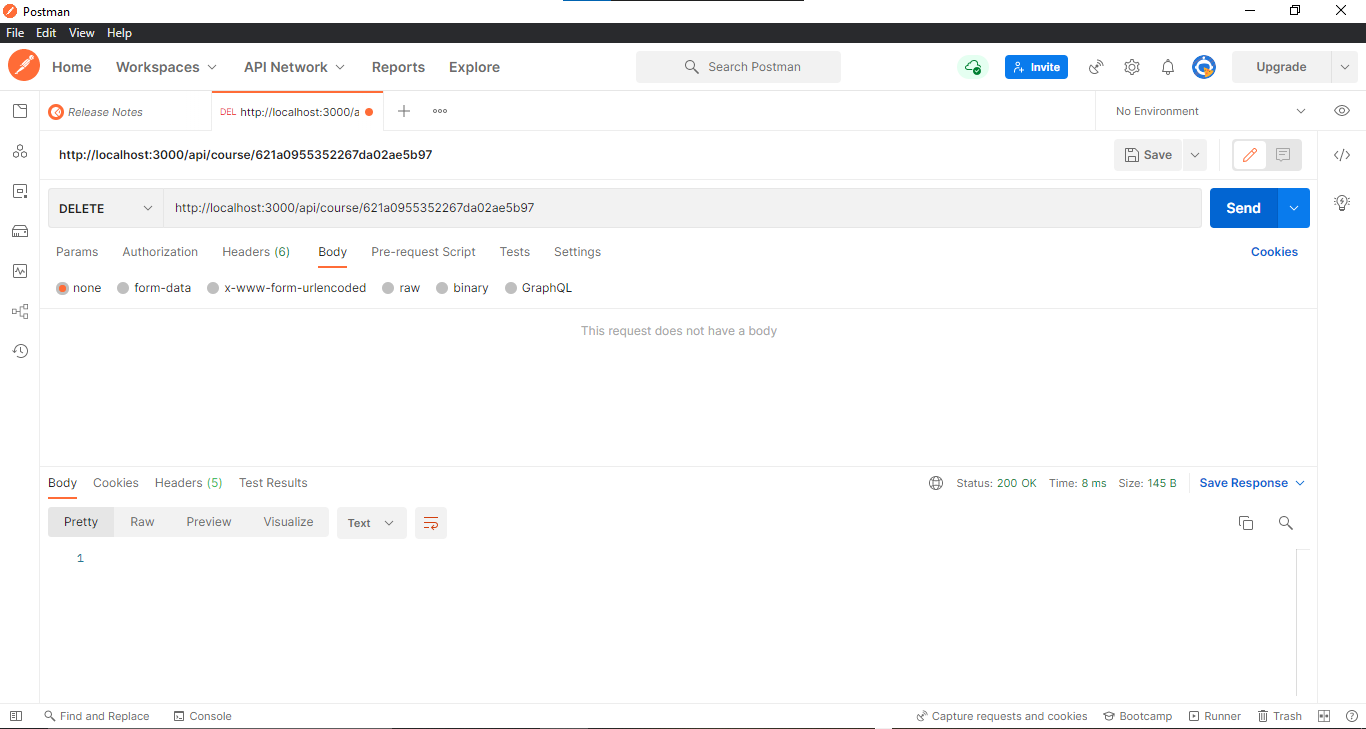






### Delete operation for courses

|  |
| --- |
| course\_routes\_api.js |
| …  router.delete("/:id",courseControllerAPI.delete);  … |
| course\_controller\_api.js |
| …  exports.delete = (req,res) => {  const id = req.path.toString().replace("/","");  courseRepository.delete(id, () => {  res.status(200).send();  });  } |





Project has been uploaded to GitHub Repository: https://github.com/zvdas/LearningManagementSystem.git

https://github.com/zvdas/LearningManagementSystem

# END OF PROJECT