

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 caret '^' 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");
});
```

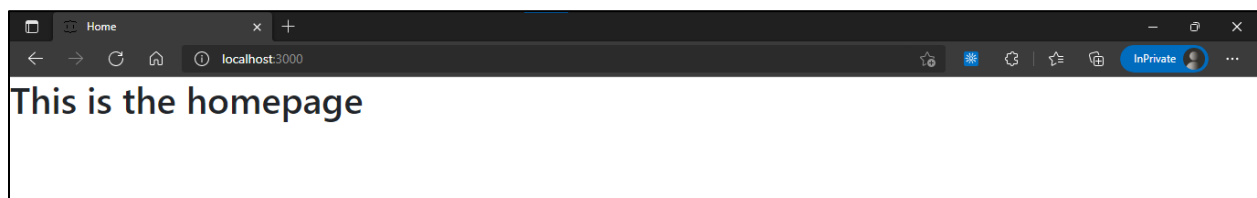
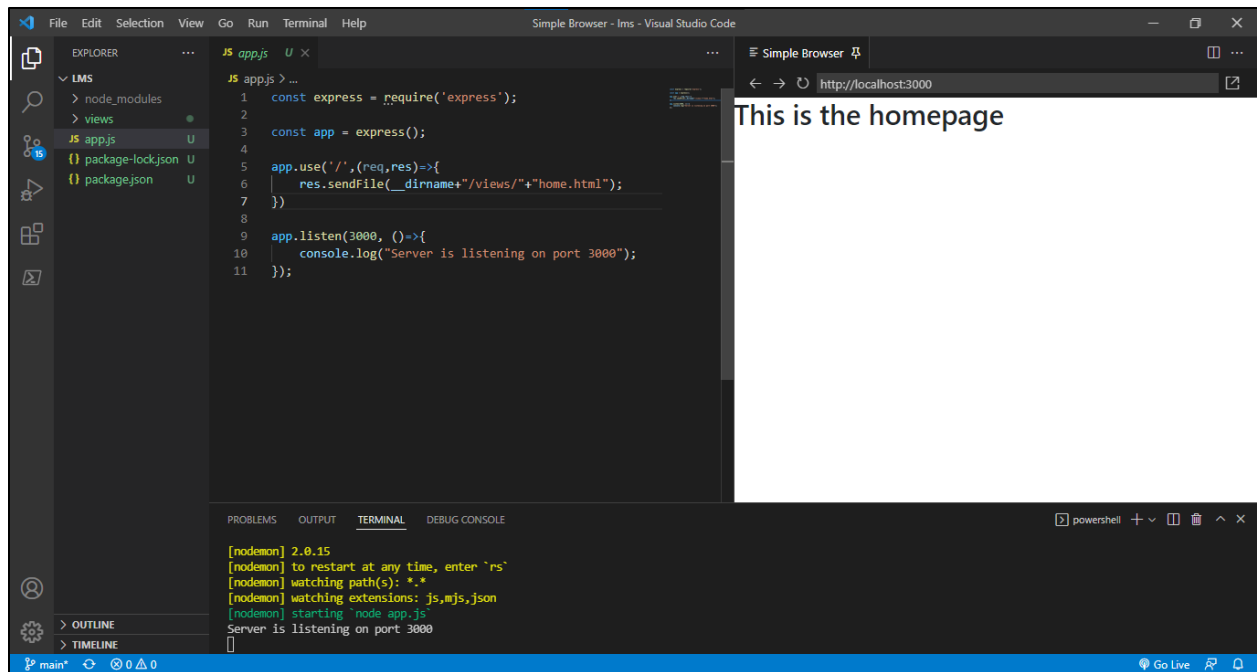
home.html

```
<!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
<pre>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')); });</pre>

```
});

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>
```

localhost:3000/user

New to EGYan? Sign up for free!

Full name Type

Email ID

Password Confirm Password

Have an account? Log in now!

Email ID






Password

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>
```


Welcome,
AAAAA

-  Dashboard
-  Manage Courses
-  Manage Lessons
-  Manage Course Material
-  Edit Profile

Logout

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>
      </div>
    </form>
  </div>
</body>
</html>
```

```

        <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>

```

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/user/edit'. The page contains two side-by-side forms. The 'Change Profile' form on the left has two input fields: 'Full name' with the value 'aa' and 'Email ID' with the value 'aa@gmail.com'. Both fields have a small blue icon to the right. Below these fields is a blue button labeled 'Change Profile'. The 'Change Password' form on the right has two input fields: 'New Password' and 'Confirm New Password'. Below these fields is a blue button labeled 'Change Password'.

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

```

Create New Course

Course Name

Category

One Liner

Duration in Hours Language

Upload Cover Photo

Description

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>
                  
                </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>

```

My Courses					
#	Photo	Course Name	Category	Duration	
1		Node.js	Web Development	1 Hours	View

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

```

Update Course

Select Course

Course Name

Category

One Liner

Duration in Hours

Language

Upload Cover Photo

Description

Update Course

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>

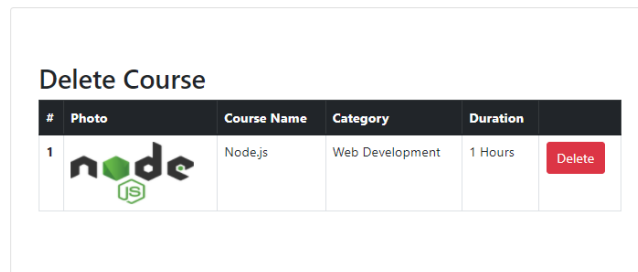
```



```

        <th scope="col"></th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <th scope="row">1</th>
        <td>
          
        </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
<pre>... const app = express(); app.set('view engine','pug'); app.set('views','views/dynamic'); app.use(bodyParser.urlencoded({extended: false})); ...</pre>
multer-config.js
<pre>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 });</pre>
course_routes.js
<pre>... const courseController = require('../controllers/course_controller'); const multerConfig = require('../configurations/multer-config'); const router = express.Router();</pre>

```

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")
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-danger", type="button",
href="/course/delete?id="+course._id) Delete
    tr

```

- [Dashboard](#)
- [New Courses](#)
- [Update Courses](#)
- [Delete Courses](#)
- [Manage Lessons](#)
- [Edit Profile](#)

My Courses

#	Photo	Course Name	Category	Duration	
1		Node.js	Web Development	1 Hours	View

- [Dashboard](#)
- [New Courses](#)
- [Update Courses](#)
- [Delete Courses](#)
- [Manage Lessons](#)
- [Edit Profile](#)

Course Name	Category	
<input type="text"/>	Web Development	
One Liner		
<input type="text"/>		
Duration in Hours	Language	Upload Cover Photo
<input type="text"/>	<input type="text"/>	<input type="button" value="Choose File"/> <input type="button" value="No file chosen"/>

[Create New Course](#)

Welcome,
AAAAA

- Dashboard
- New Courses
- Update Courses
- Delete Courses
- Manage Lessons
- Edit Profile

Logout

Change Profile

Full name

Email ID

Change Profile

Change Password

New Password

Confirm New Password

Change Password

New to EGyan? Sign up for free!

Full name

Type

Email ID

Password

Confirm Password

Register Now!

Have an account? Log in now!

Email ID

Password

Login

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.b
ody._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');

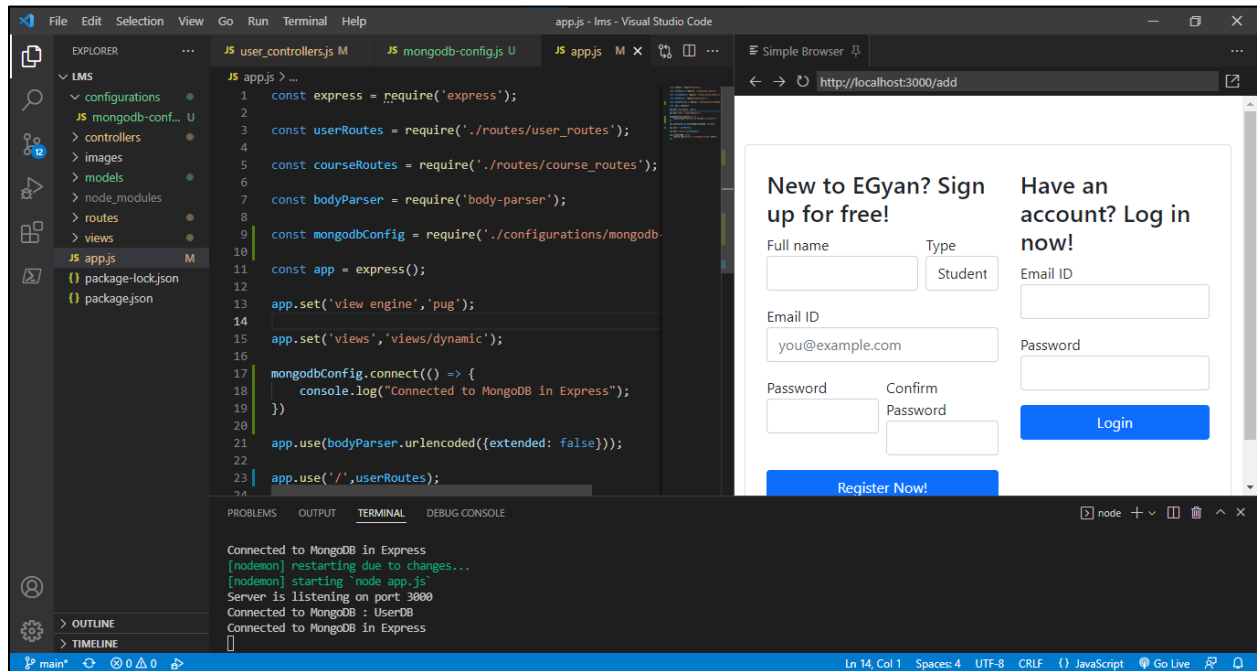
mongodbConfig.connect(() => {
  console.log("Connected to MongoDB in Express");
})

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');

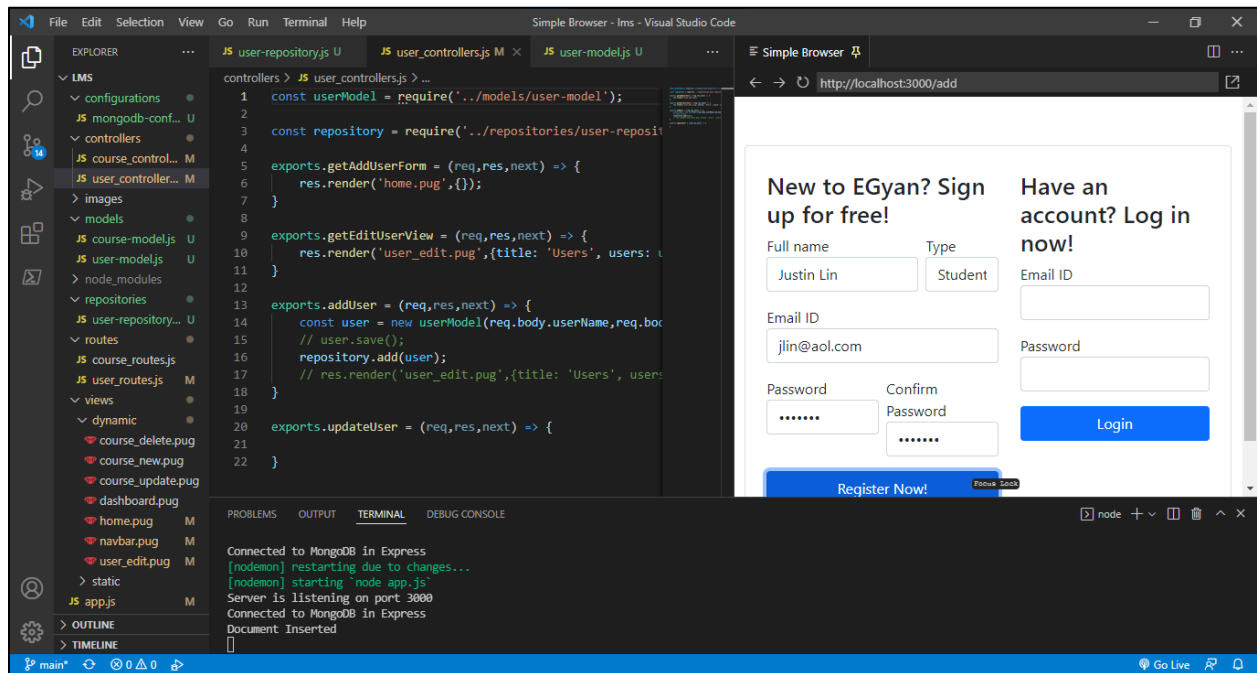
const repository = require('../repositories/user-repository');
...
exports.addUser = (req,res) => {
    if(Object.values(req.body).length===4) {
        userRepository.getAll((users) => {
            if(JSON.stringify(users).includes(req.body.userEmail)) {
                res.render('home.pug',{ title: '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.getByd(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.' });
}
});
}
};
...

```



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
MongoDB shell version v5.0.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1434698e-f441-45e4-be3f-8d4df0725685") }
MongoDB server version: 5.0.6
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
---
    2022-02-13T11:00:09.176+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
    Enable MongoDB's free cloud-based monitoring service, which will then receive and display
    metrics about your deployment (disk utilization, CPU, operation statistics, etc).

    The monitoring data will be available on a MongoDB website with a unique URL accessible to you
    and anyone you share the URL with. MongoDB may use this information to make product
    improvements and to suggest MongoDB products and deployment options to you.

    To enable free monitoring, run the following command: db.enableFreeMonitoring()
    To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
> show dbs;
admin    0.000GB
config   0.000GB
local    0.000GB
userDB    0.000GB
> use userDB;
switched to db userDB
> show collections;
users
> db.users.find();
{ "_id" : ObjectId("6208a4f875b51e329628aee7"), "userName" : "Justin Lin", "userType" : "Student", "userEmail" : "jlin@aol.com", "userPassword" : "jlin123" }
>
```

Read Operation for users

user-repository.js
<pre>... exports.getAll = (callback) => { const collection = database.getCollection(); collection.findOne() .then((users) => { return callback(users); }); }</pre>
user_controllers.js
<pre>... 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}); }); } ...</pre>

<pre> 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} ... </pre>
<pre> 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) </pre>

The screenshot displays a user interface with a dark sidebar on the left and a light main content area on the right. The sidebar contains a welcome message 'Welcome, Justin Lin' and a list of navigation links: Dashboard, New Courses, Update Courses, Delete Courses, Manage Lessons, and Edit Profile. A 'Logout' button is at the bottom of the sidebar. The main content area features two side-by-side forms. The 'Change Profile' form has input fields for 'Full name' (containing 'Justin Lin') and 'Email ID' (containing 'jlin@aol.com'), with a 'Change Profile' button below. The 'Change Password' form has input fields for 'New Password' and 'Confirm New Password', with a 'Change Password' button below.

Update Operation for users

<pre> user-model.js const users = []; module.exports = class userModels{ constructor(userName,userType,userEmail,userPassword,id){ this.userName = userName; this.userType = userType; this.userEmail = userEmail; </pre>

```
    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();
  });
}
```



```

user-repository.js
6 collection.insertOne({ userName: user.userName, userType: user.userType })
7   .then(() => {
8     console.log("Document Inserted");
9   })
10 }
11
12 exports.getAll = (callback) => {
13   const collection = database.getCollection();
14   collection.findOne()
15   .then((users) => {
16     return callback(users);
17   });
18 }
19
20 exports.update = (user, callback) => {
21   const collection = database.getCollection();
22   collection.findOneAndUpdate({ _id: ObjectId(user._id) }, { $set: {
23     userName: user.userName,
24     userType: user.userType
25   } })
26   .then(() => {
27     console.log("Document Updated");
28     return callback();
29   });
30 }

```

```

user_controllers.js
9 exports.getEditUserView = (req, res, next) => {
10   repository.getAll((users) => {
11     res.render('user_edit.pug', { title: 'Users', users: users });
12   });
13 }
14
15 exports.addUser = (req, res, next) => {
16   const user = new userModel(req.body.userName, req.body.userType);
17   repository.add(user);
18   repository.getAll((users) => {
19     res.render('user_edit.pug', { title: 'Users', users: users });
20   });
21 }
22
23 exports.updateUser = (req, res, next) => {
24   const user = new userModel(req.body.userName, req.body.userType);
25   repository.update(user, () => {
26     repository.getAll((users) => {
27       res.render('user_edit.pug', { title: 'Users', users: users });
28     });
29   });
30 }

```

```

Terminal
Connected to MongoDB in Express
[nodemon] restarting due to changes...
[nodemon] starting node app.js
Server is listening on port 3000
Connected to MongoDB in Express
Document Updated

```

```

C:\Program Files\MongoDB\Server\3.0\bin>mongo.exe
config 0.000GB
local 0.000GB
userDB 0.000GB
> use userDB;
switched to db userDB
> show collections;
Users
> db.Users.insert({
  "id": ObjectId("6208a4f875b51e329628aee7"), "userName": "Justin Lin", "userType": "Student", "userEmail": "jlin@aol.com", "userPassword": "jlin123" });
{
  "id" : ObjectId("6208a4f875b51e329628aee7"),
  "userName" : "Justin Lin",
  "userType" : "Student",
  "userEmail" : "jlin@aol.com",
  "userPassword" : "jlin123"
}
> db.Users.findOne();
{
  "id" : ObjectId("6208a4f875b51e329628aee7"), "userName": "Justin Lin", "userType": "Student", "userEmail": "jlin@aol.com", "userPassword": "jlin123" }
> db.Users.insert({
  "id": ObjectId("6208a4f875b51e329628aee7"), "userName": "Justin Ling Chian Wing", "userType": "Student", "userEmail": "jlinling@gmail.com", "userPassword": "123" });
{
  "id" : ObjectId("6208a4f875b51e329628aee7"),
  "userName" : "Justin Ling Chian Wing",
  "userType" : "Student",
  "userEmail" : "jlinling@gmail.com",
  "userPassword" : "123"
}
> db.Users.findOne();
{
  "id" : ObjectId("6208a4f875b51e329628aee7"),
  "userName" : "Justin Ling Chian Wing",
  "userType" : "Student",
  "userEmail" : "jlinling@gmail.com",
  "userPassword" : "123"
}

```

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();
    });
};
};

```

The screenshot shows a web application interface. On the left is a dark sidebar with the text 'Welcome, Justin' and a 'Logout' button. The main area is titled 'Delete Users' and contains a table with the following data:

#	Name	Type	Email	Password	
1	Jeremy Wayans	Student	jwayans@aol.com	jwayans123	Delete
2	Justin Ling Chian Wing	Student	justinling@gmail.com	jlin123	Delete

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] });
}

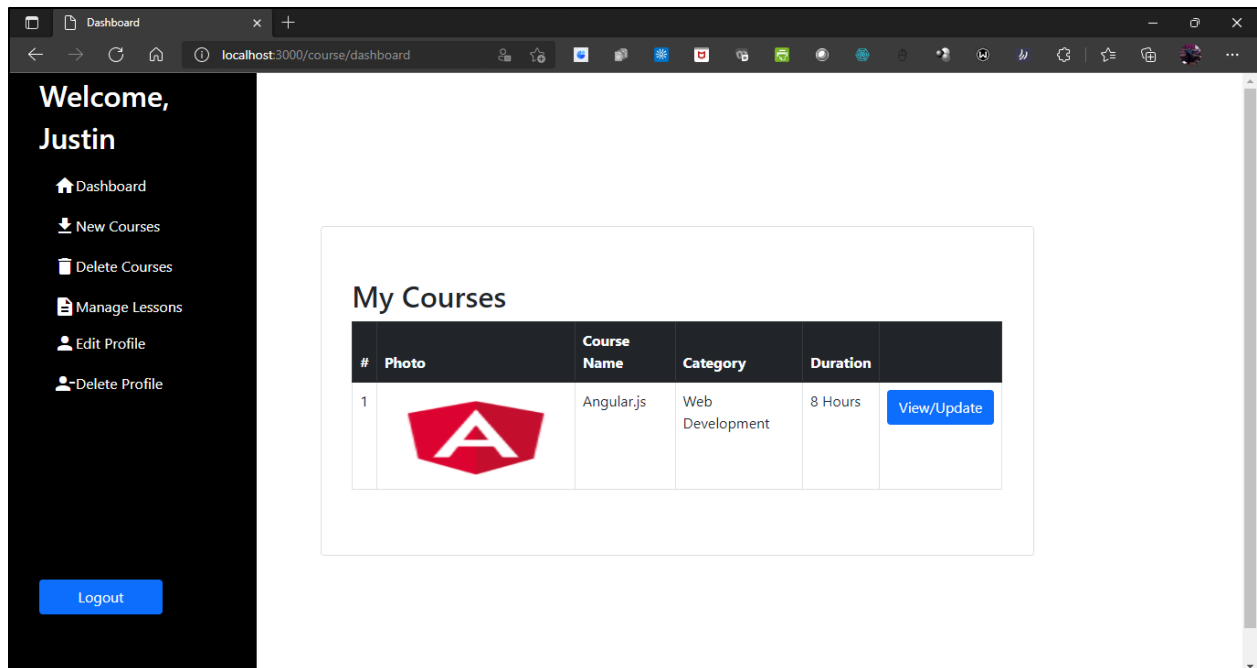
```



```

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

```

The screenshot displays a web browser window with the URL `localhost:3000/course/update?id=621...`. The page is titled "Update Course". On the left, a dark sidebar shows a "Welcome, Justin" message and a list of navigation options: Dashboard, New Courses, Delete Courses, Manage Lessons, Edit Profile, and Delete Profile, with a "Logout" button at the bottom. The main content area contains the "Update Course" form. The form has the following fields: "Course Name" (Angular.js), "Category" (Web Development), "One Liner" (Learn to Create Single Page Applications with Angular.js), "Duration in Hours" (8 Hours), "Language" (English (JavaScript)), and "Upload Cover Photo" (Choose File, No file chosen). Below these is a CKEditor for the "Description" field, which contains the text "Learn to Create Single Page Applications with Angular.js". At the bottom right of the form is a blue "Update Course" button.

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] });
            });
        })
    }
};
```

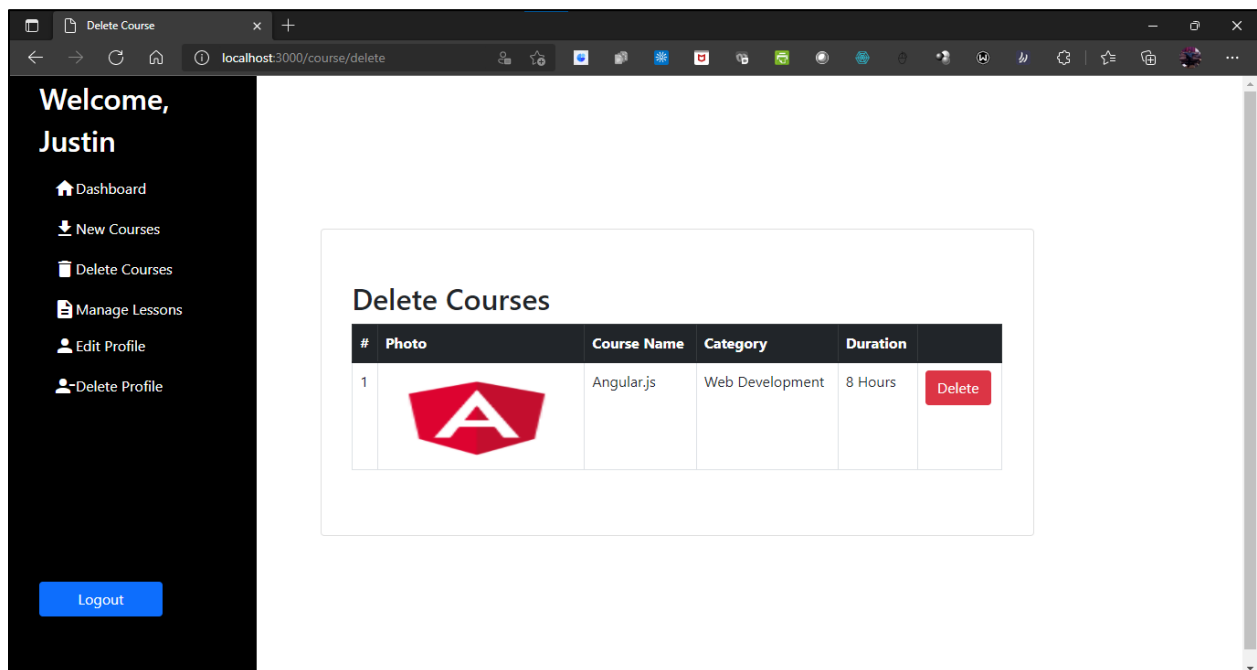
course_delete.pug

```
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")
```

```

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-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);

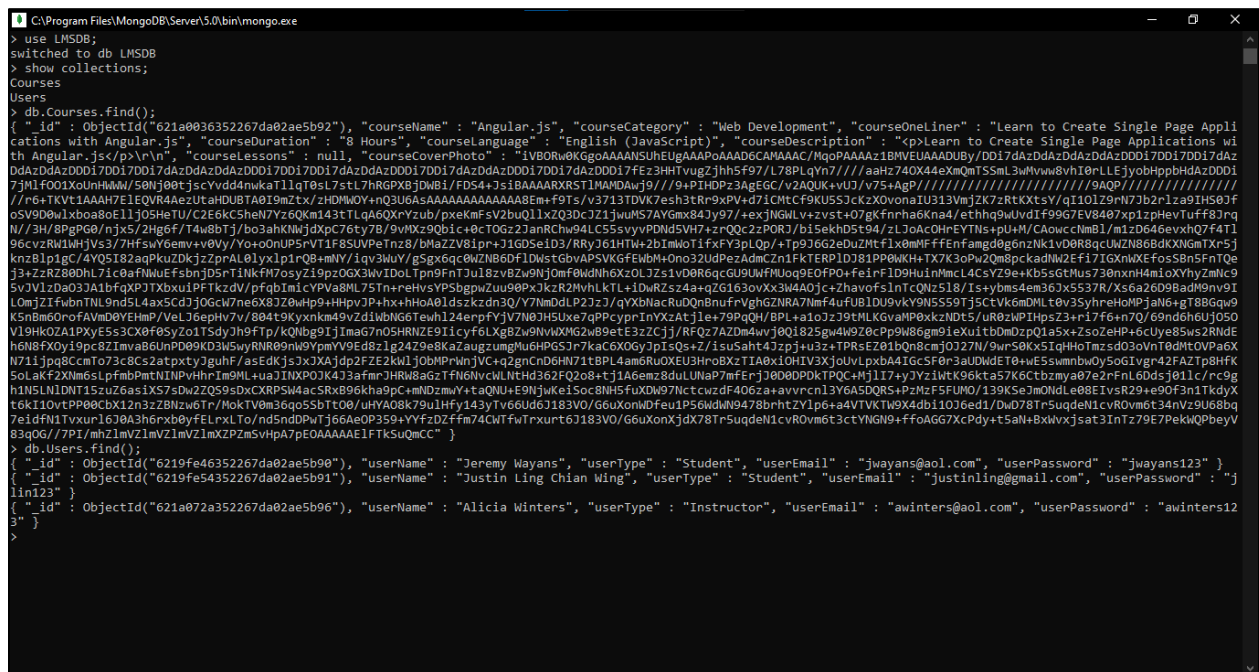
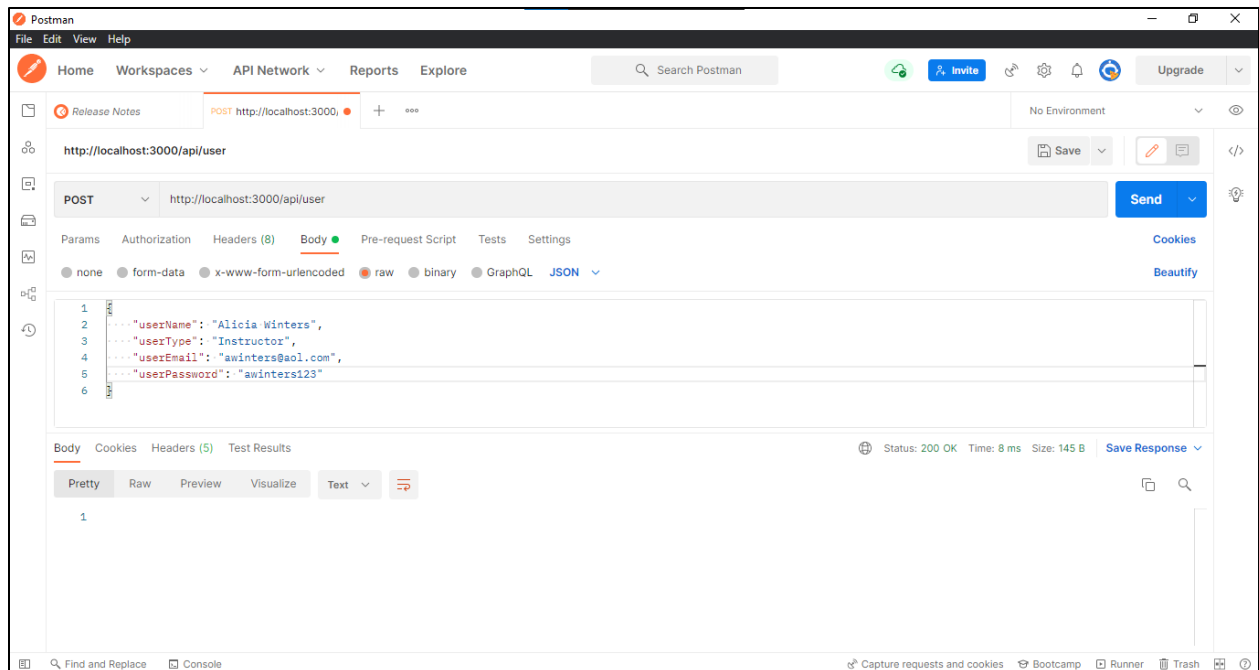
module.exports = router;
```

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();
};
```



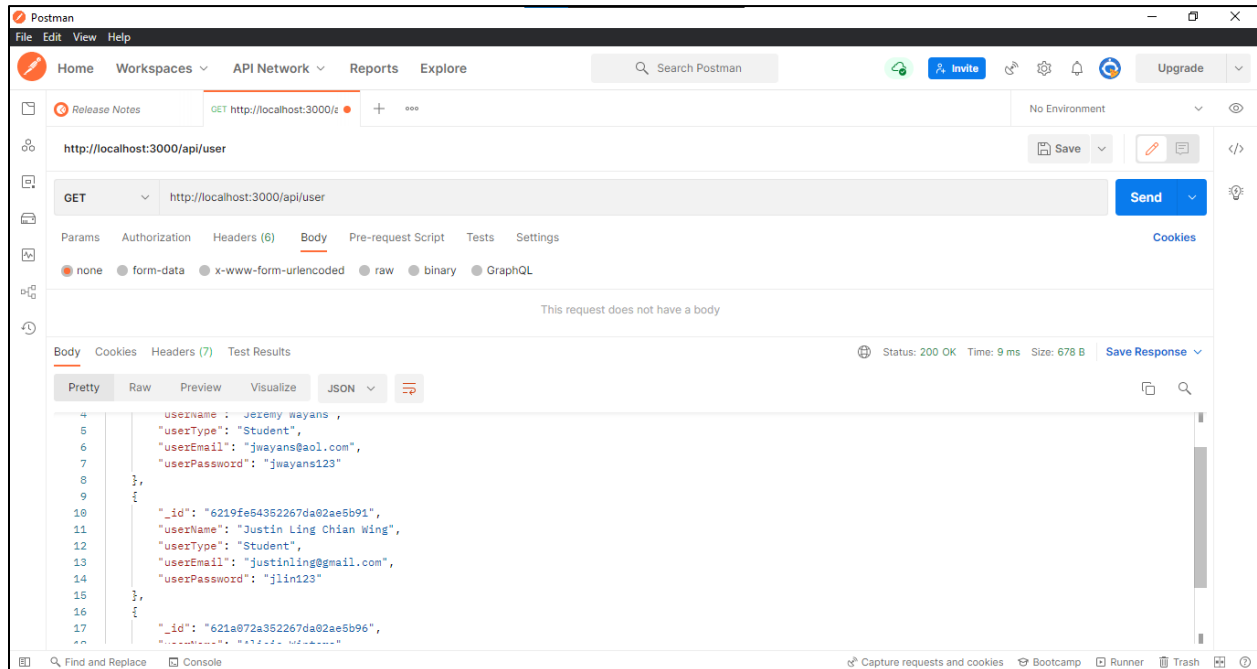
Read operation for users

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

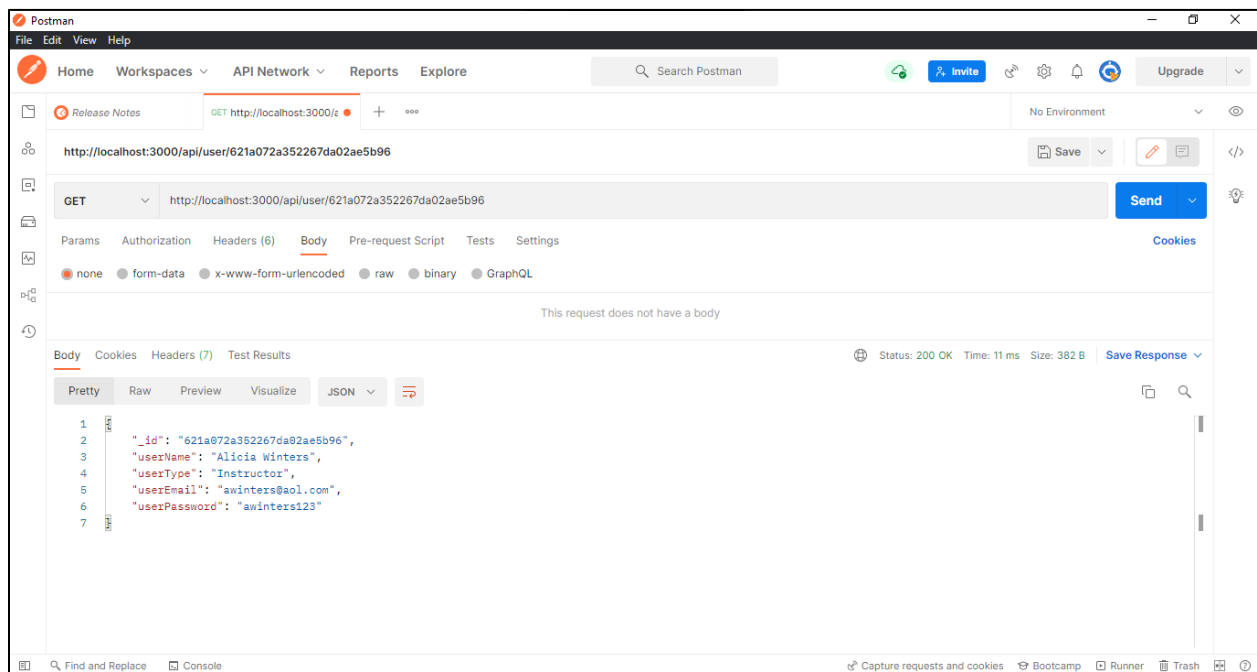
```

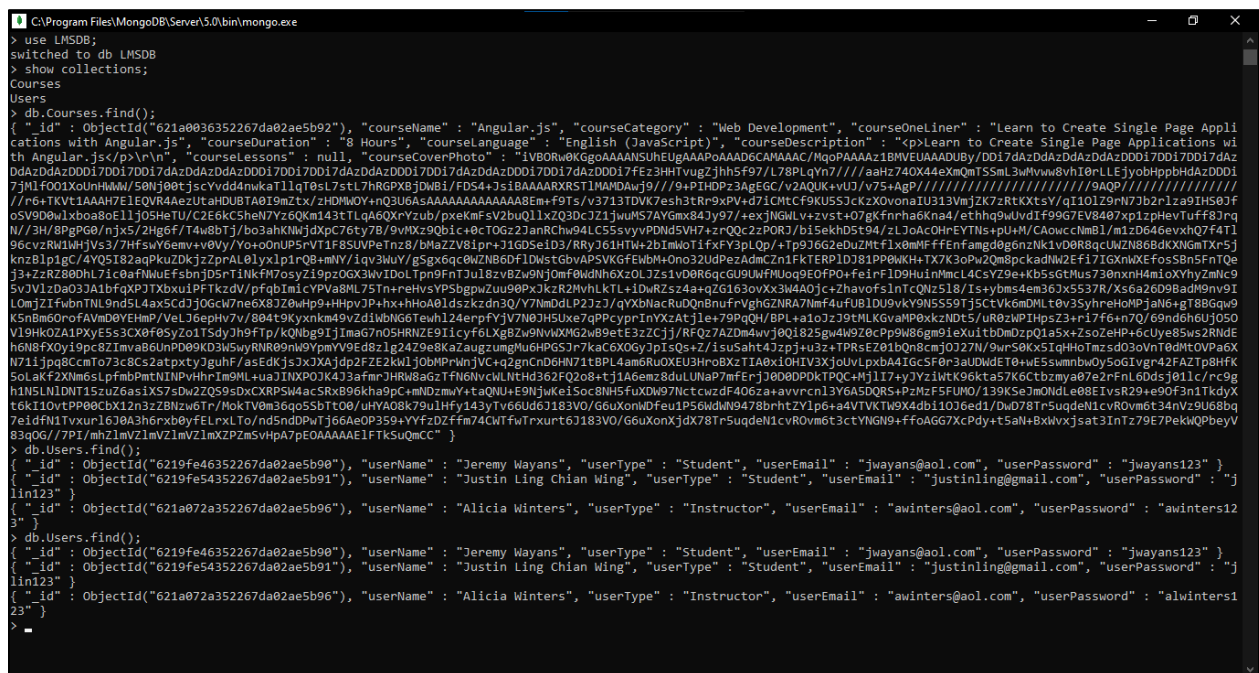
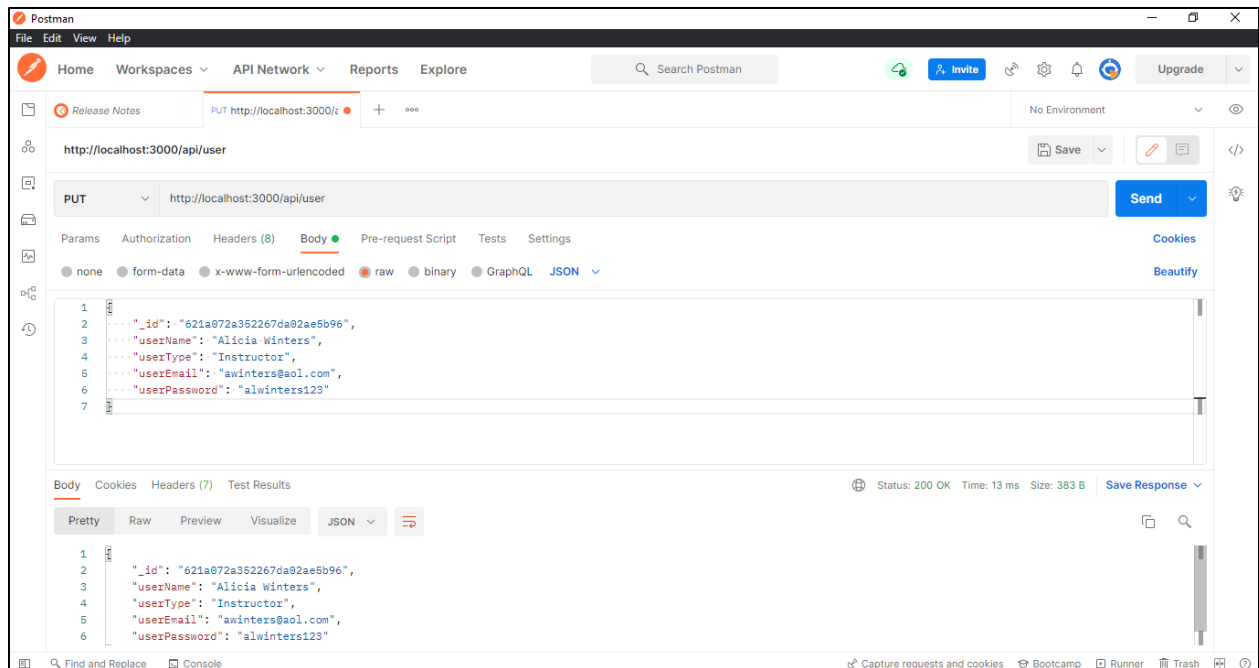
...
exports.getById = (req,res) => {
  const id = req.path.toString().replace("/", "");
  userRepository.getById(id, (user) => {
    if(user){
      res.status(200).send(user);
    }else{
      res.status(404).send("Error");
    }
  });
};

exports.update = (req,res) => {

```

```
const user = new
userModel(req.body.userName, req.body.userType, req.body.userEmail, req.body.userPassword, req.b
ody._id);
  userRepository.update(user, () => {
    userRepository.getByld(user._id, (userResult) => {
      res.status(200).send(userResult);
    });
  });
};
```

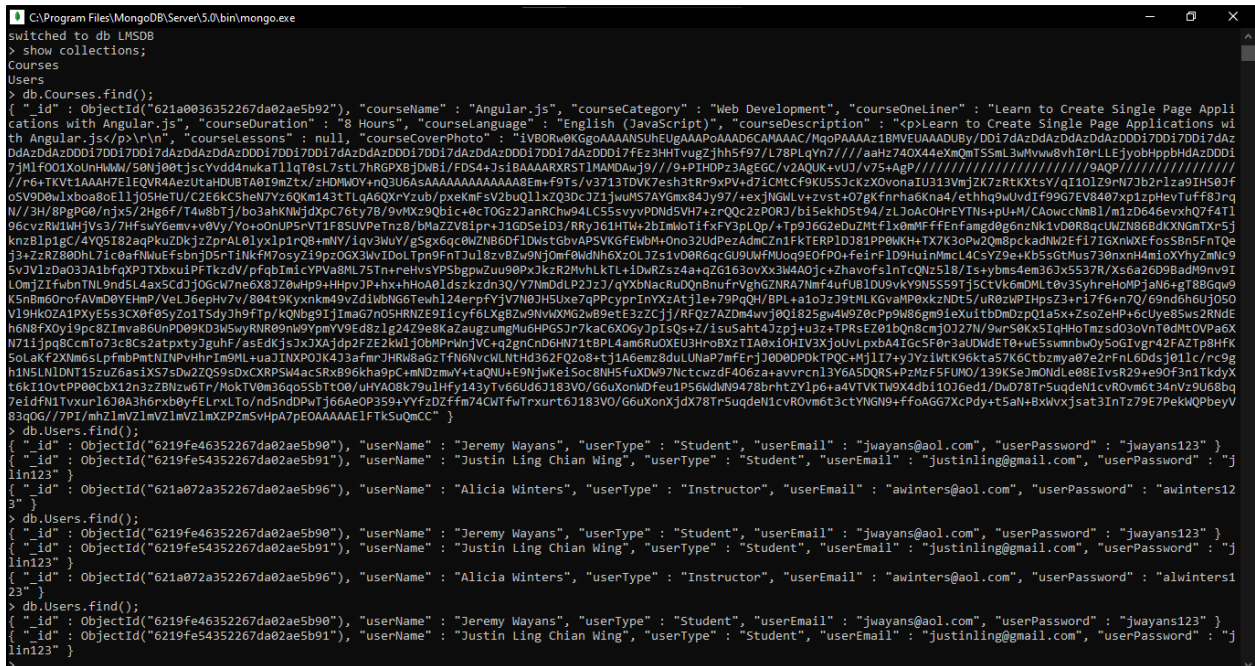
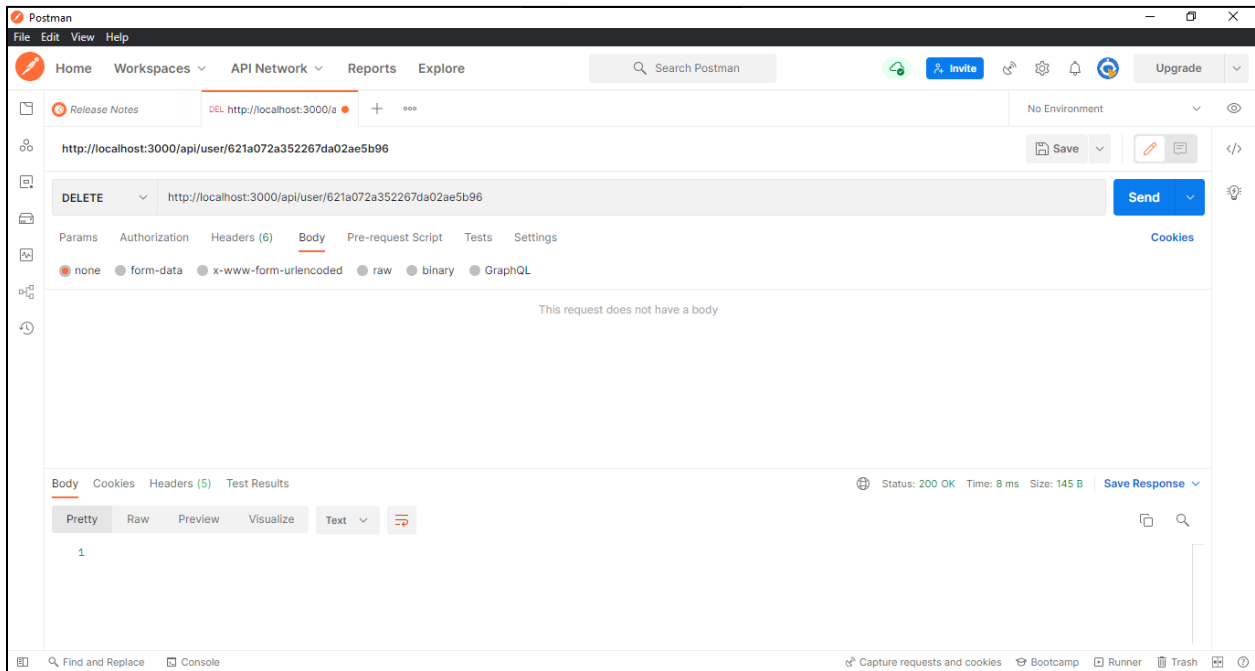




Delete operation for users

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);

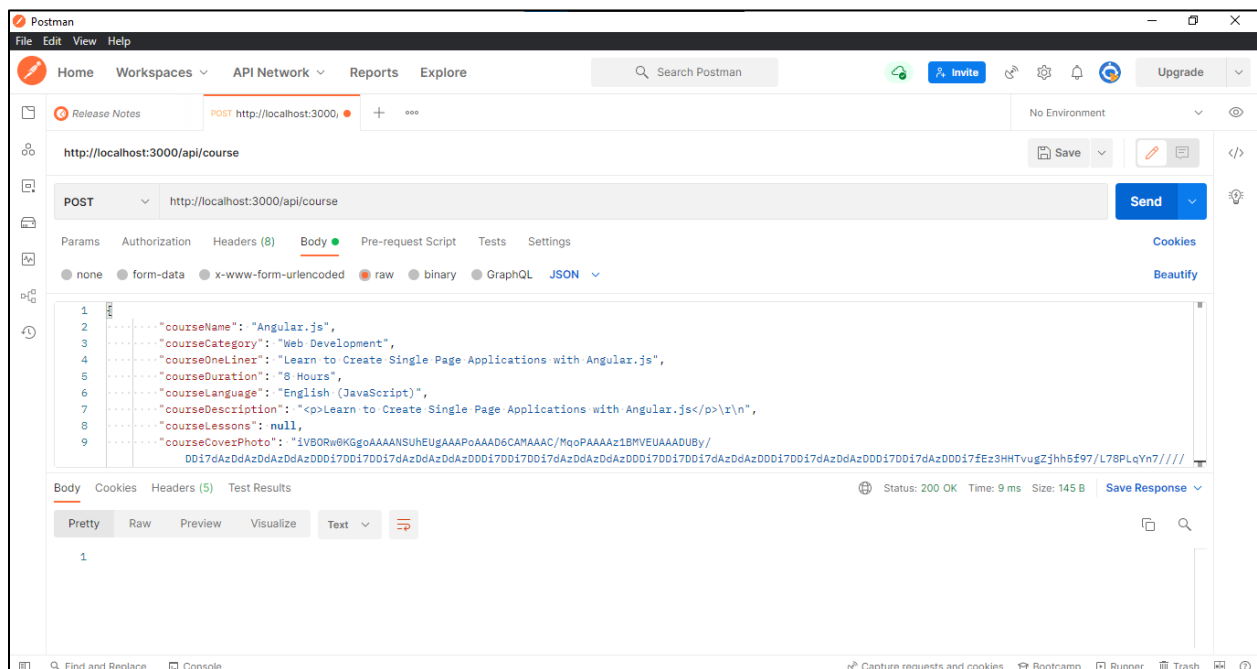
module.exports = router;
```

```
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); }); }



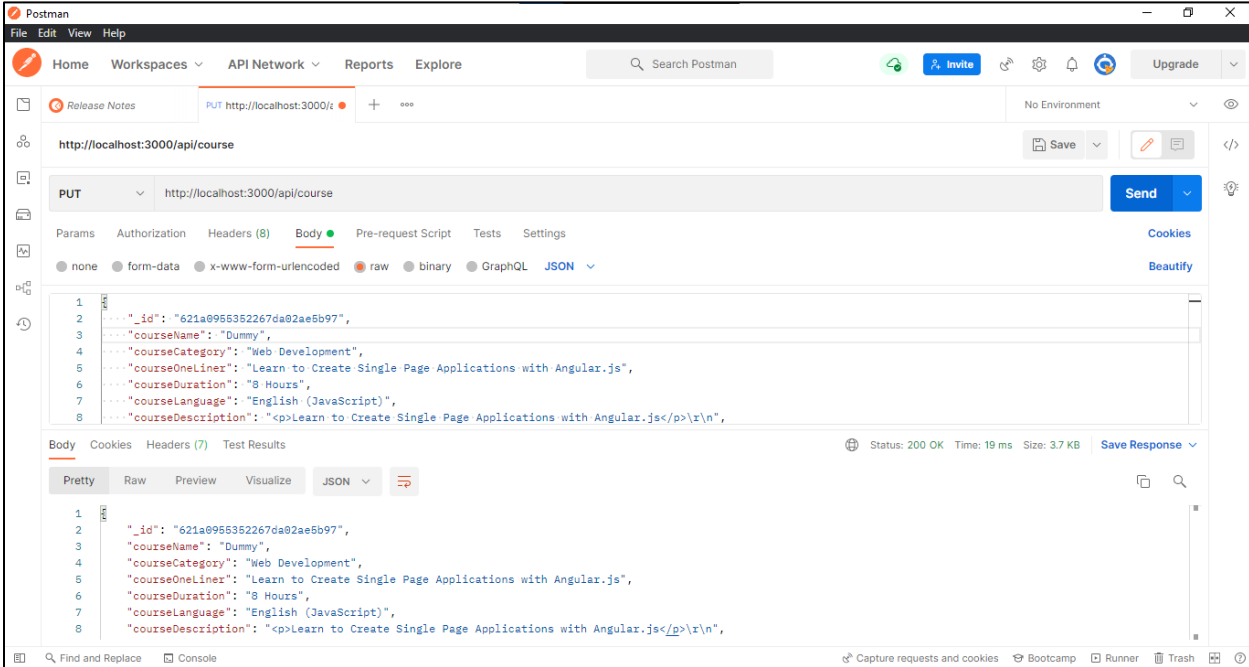
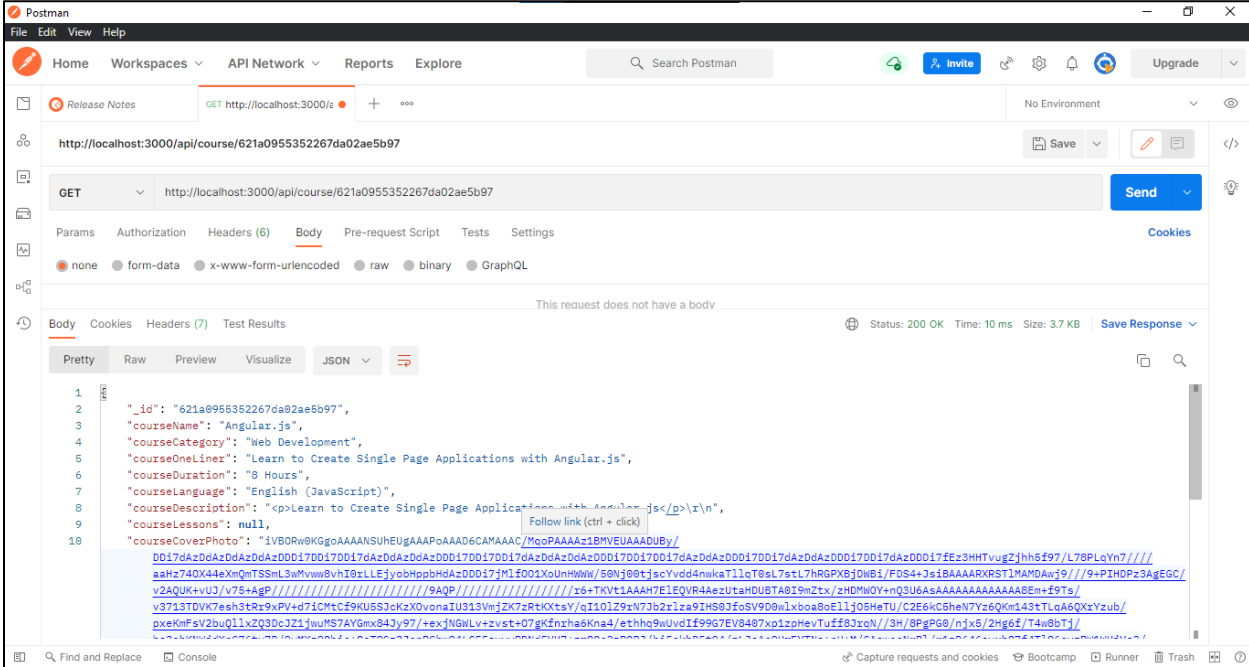
```
...
router.get("/:id",courseControllerAPI.getById);

router.put("/",courseControllerAPI.update);
...
```

```
...
exports.getByld = (req,res) => {
  const id = req.path.toString().replace("/", "");
  courseRepository.getByld(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.getByld(course._id, (courseResult) => {
      res.status(200).send(courseResult);
    });
  });
}
```

```
});
}
```





Delete operation for courses

course_routes_api.js
... router.delete("/:id",courseController.delete); ...
course_controller_api.js
... exports.delete = (req,res) => { const id = req.path.toString().replace("/", ""); courseRepository.delete(id, () => { res.status(200).send(); }); }

