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

1. **Business Requirement - Overview**To develop the Learning management application with features given below:
2. **Front-end Component:** There are different pages in the application that have to be created i.e. Home page ,Registration page and dashboard page etc.,.
3. **API (Application Programming Interface) Component:** 
   1. **API Component (** /api/login **)**: Implement REST endpoints POST.
   2. **API Component (** /api/register **)**: Implement REST endpoints GET, POST
   3. **API Component (** /api/createCourse **)**: Implement REST endpoints GET, POST
   4. **API Component (** /api/updateCourse **)**: Implement REST endpoints GET, POST
   5. **API Component (** /api/deleteCourse **)**: Implement REST endpoints GET, POST
   6. **API Component (** /api/editProfile **)**: Implement REST endpoints GET, POST
   7. **API Component (** /api/changeProfile **)**: Implement REST endpoints GET, POST
   8. **API Component (** /api/changePassword **)**: Implement REST endpoints GET, POST
4. **Schema Component:** Create collections **user**, **course** using MongoDB or MySQL.
5. **Technical Specifications**

**2.1. Software and application details:**

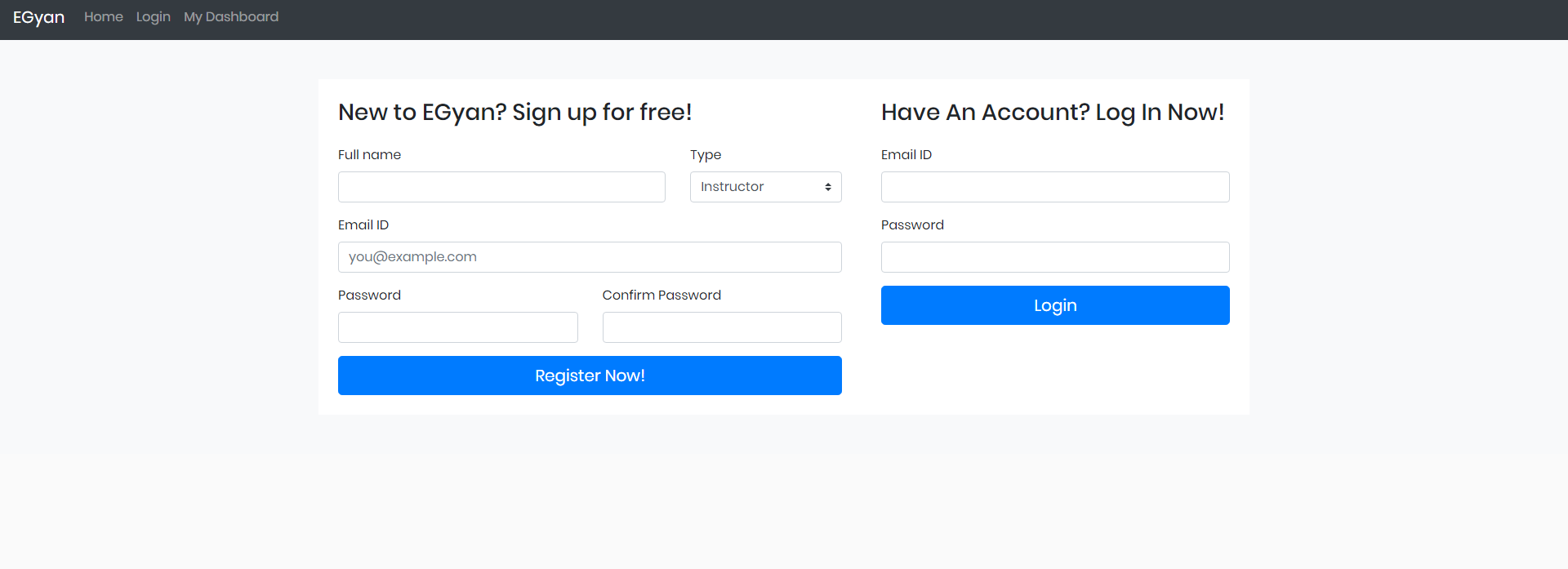
**2.2. UI (User Interface) - Design wireframes**

The UI wireframes with the required layouts are as follows:

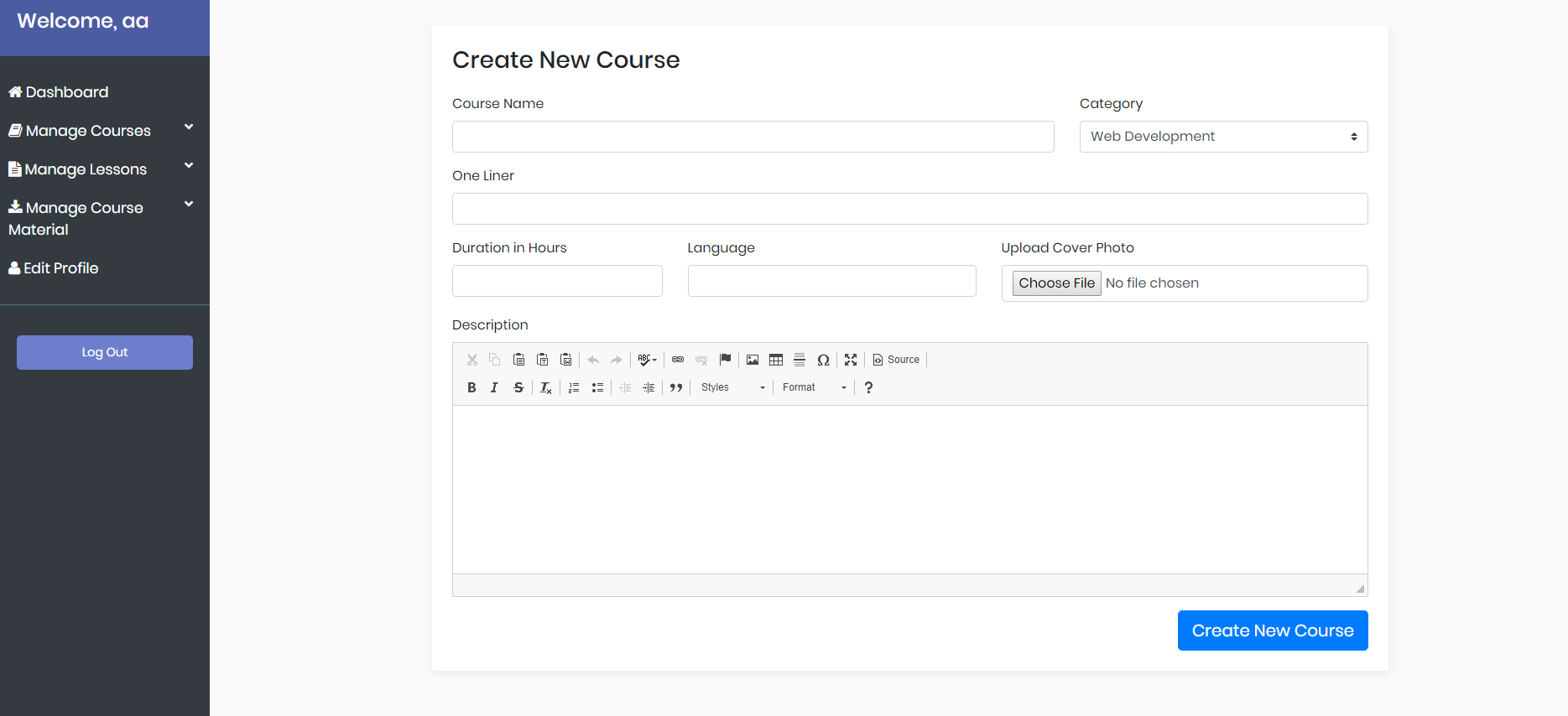
**NOTE:** The candidate will not be evaluated based on the UI design (layout, color, formatting, and so on). The candidate is free to have a basic UI with all the required UI components (input fields, buttons, labels, and so on ).

Following screenshot represents the registration page view:

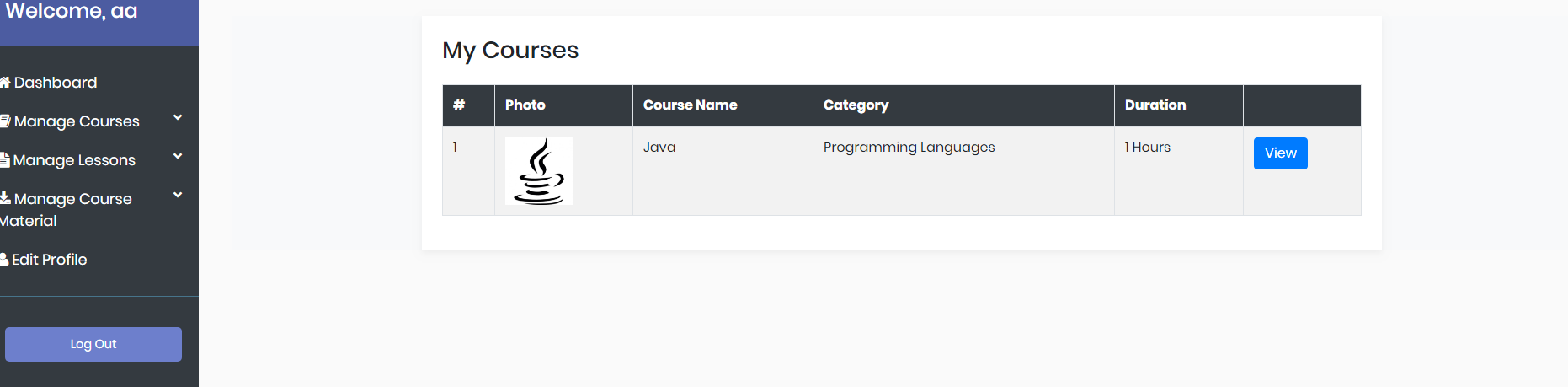
* After the build of base code is completed:

  
After the features are developed, the page looks like this:

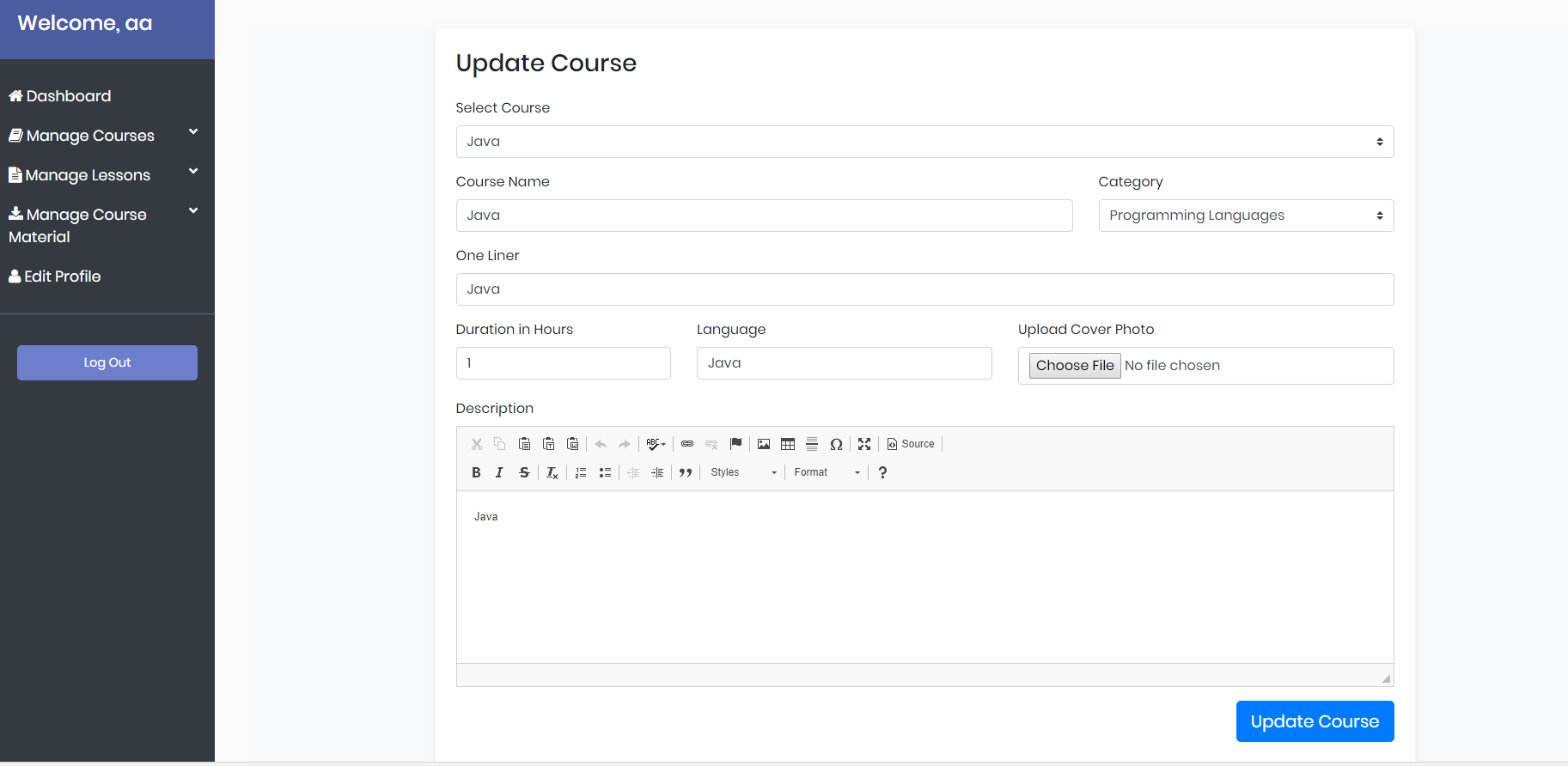
**Create New Course page**



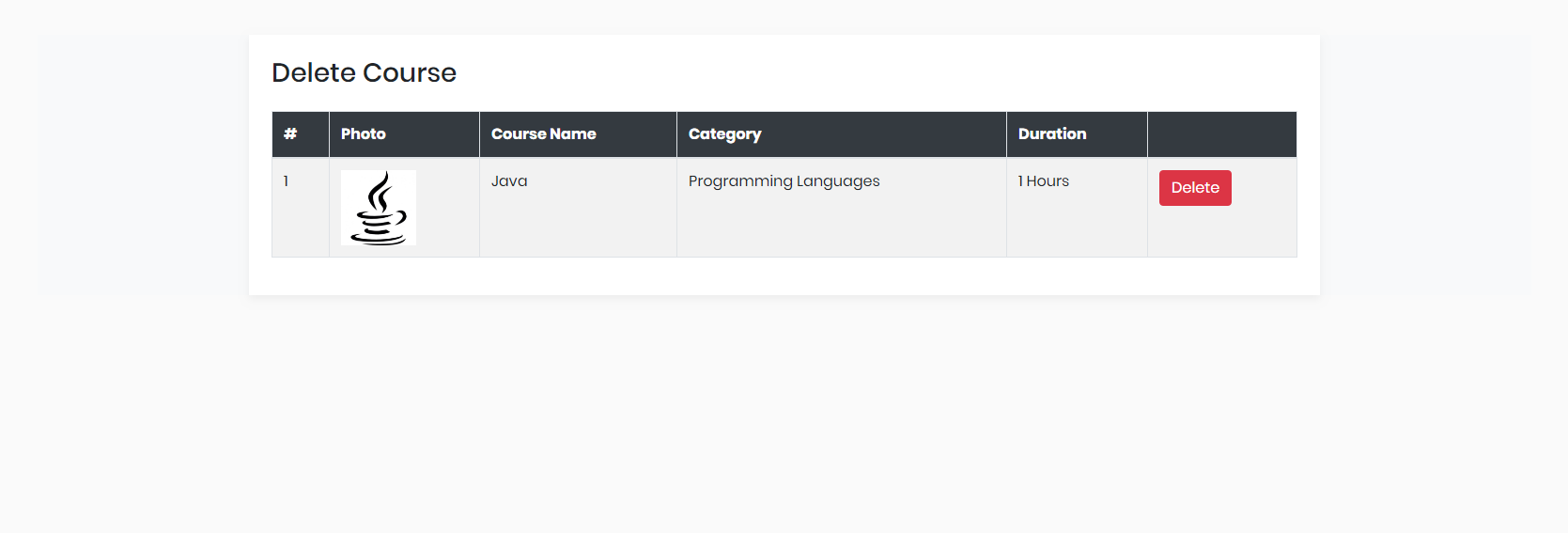
**Dashboard page**



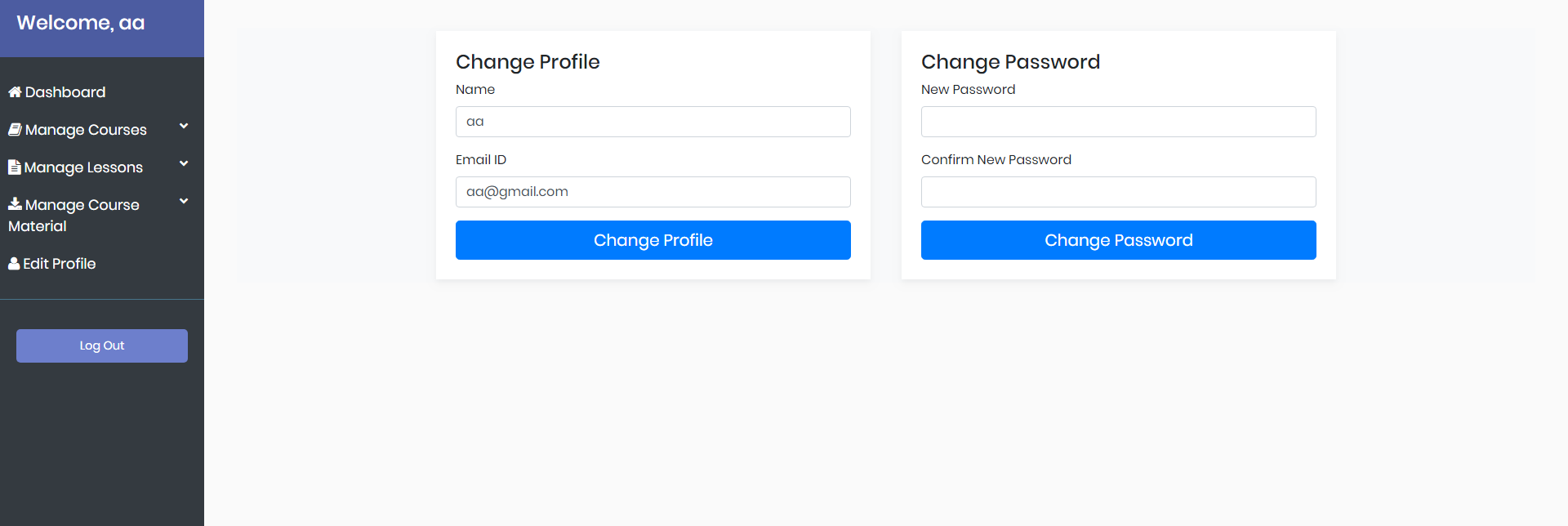
**Update Course page**



**Delete Course page**



**Edit user page**



1. **Application features**

The application should have the following set of built-in features:

**Front-end Component:**

* **Home page**:
  + - The Home page has a button where user can on click it will take user to registration page
* **Registration page**:
  + - In registration page user can register and sign in
    - The dashboard page contains following components: Manage course, Manage Lesson, Manage Course Material, Edit Profile

**API Component:**Manage Lesson, Manage Course Material.

1. **Project deliverables**

Below are the deliverables:

**Front-end Component** :HTML and CSS code for the **Registration** and **login page**,

**Manage Course** , **edit Profile** is provided ,Implement the missing logic using

JavaScript

**fileNames**:

register.ejs

createCourse.ejs

updateCourse.ejs

deleteCourse.ejs

editProfile.ejs

index.js

Make the given files to make the application work properly

The user can register in two ways:

1) As an instructor

2) As a student

Registration and login process should be working

Course creation, update, deletion should be implemented accordingly

Implement the following logics

1. Implement login using the Post method (send Email id and password)

Example request body should be in:

**{emailId: "test@gmail.com", password: "test"}**

1. User registration should be implemented (take the required details and send post request for user registration)

Example request body should be in:

**{“name”: "test", “type”: "1", “emailId”: "test@gmail.com", “password”: "aaa"}**

Type 1 for instructor:

**{“name”: "test", “type”: "2", “emailId”: "test@gmail.com", “password”: "aaa"}**

Type will be 2 for student:

1. Implement create course functionality which is a post method refer stubs for further information:

**{**

**"name": "Java",**

**"category": "Programming Languages",**

**"oneLiner": "Java",**

**"duration": "1",**

**"language": "Java",**

**"description": "%3Cp%3EJava%20..%3C%2Fp%3E%20",**

**"lessons": [],**

**“photo”:”photo blob text”**

**}**

1. Implement update course functionality which is a post method refer stubs for further information:

**{**

**"id": "5b6c5a67e8e0a03b40e584e7",**

**"name": "Java",**

**"category": "Programming Languages",**

**"oneLiner": "Java",**

**"duration": "1",**

**"language": "Java",**

**"description": "%3Cp%3EJava%3C%2Fp%3E%20",**

**"lessons": [],**

**“photo”:”blob data”**

**}**

1. Implement delete course functionality which is a post method refer stubs for further information.
2. Implement Javascript logic for editProfile.

## 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.

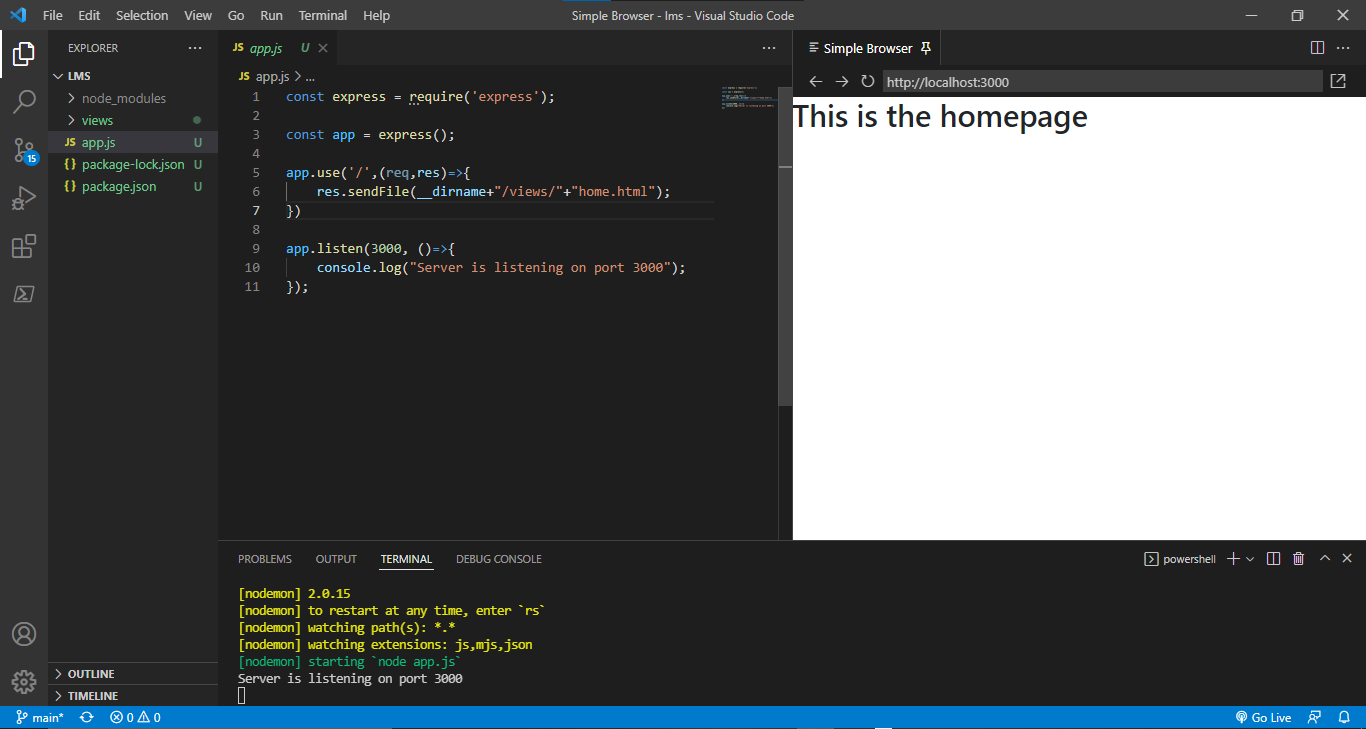
Start Express on localhost:3000

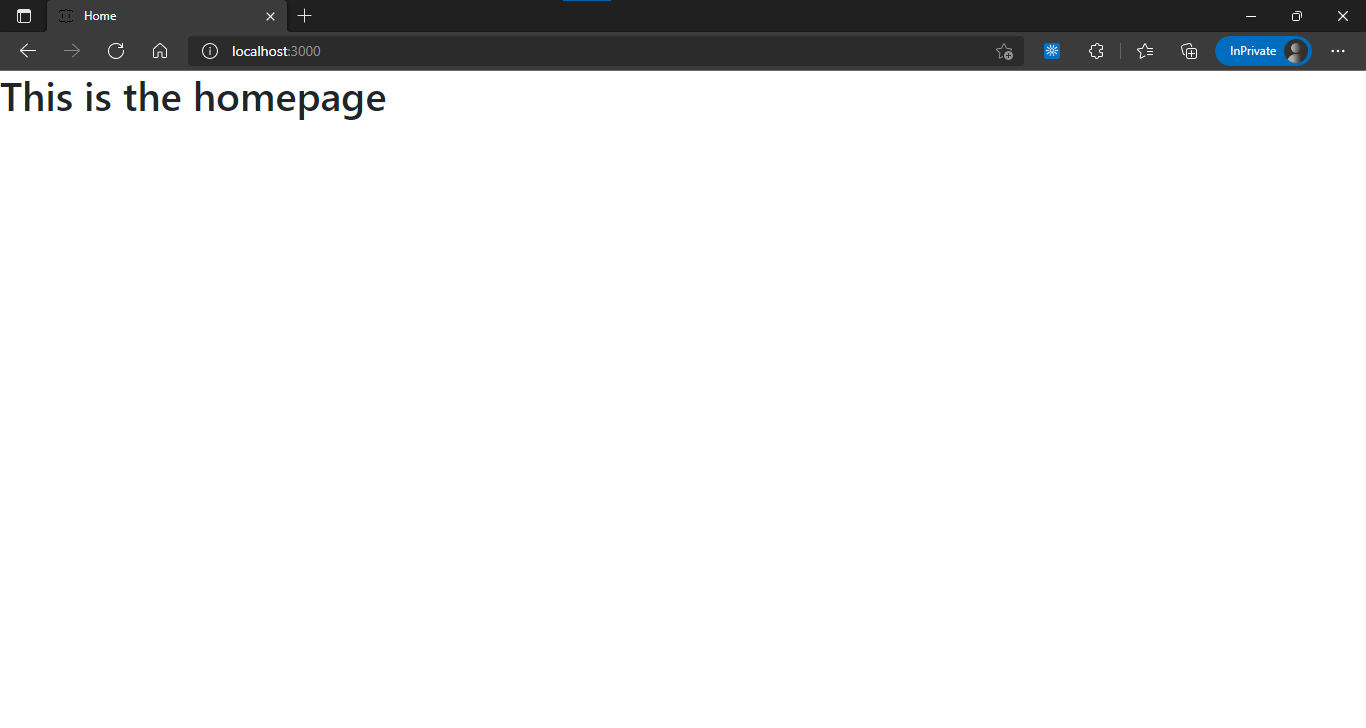
|  |
| --- |
| app.js |
| const express = require('express');  const app = express();  app.use('/',(req,res)=>{  res.sendFile(\_\_dirname+"/views/"+"home.html");  })  app.listen(3000, ()=>{  console.log("Server is listening on port 3000");  }); |
| home.html |
| <!DOCTYPE html>  <html lang="en">  <head>  <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('/', (req,res)=>{  res.sendFile(path.join(\_\_dirname,'../','views','home.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 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 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">  </head>  <body>  <div class="container">  <div class="row justify-content-center">  <div class="col align-self-center">  <form action="">  <div class="row">  <div class="col-8">  <div class="card" style="margin: 10%;">  <h1>New to EGyan? Sign up for free!</h1>  Full name<input type="text" name="name" style="width: 60%; float: left;"/>  Type<input type="text" name="type" style="width: 20%; float: right;" />  Email ID<input type="text" name="email" placeholder="you@example.com"/>  Password<input type="text" name="type" style="width: 40%;" />  Confirm Password<input type="text" name="type" style="width: 40%;" />  <button class="btn btn-primary" type="submit" name="register">Register Now!</button>  </div>  </div>  <div class="col-4">  <div class="card">  <h1>Have an account? Log in now!</h1>  Email ID<input type="text" name="email" />  Password<input type="text" name="type" />  <button class="btn btn-primary" type="submit" name="login">Login</button>  </div>  </div>  </div>  </form>  </div>  </div>  </div>  </body>  </html> |