CSE3999 - Technical Answers for Real World Problems (TARP)

Project Report

FRED

By

18BCE1041	**Keshav**
18BCE1207	**Saurabh Mohata**
18BCE1330	**Aditya Kumar**

B.Tech. Computer Science and Engineering

Submitted to

Dr. Malathi G.

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

June 2021

DECLARATION

We hereby declare that the report titled "**FRED"** submitted by us to VIT Chennai is a record of bona-fide work undertaken by us under the supervision of **Dr. Malathi G.**, School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

KESHAV		
Keshav	**Saurabh Mohata**	**Aditya Kumar**
18BCE1041	**18BCE1207**	**18BCE1330**

CERTIFICATE

Certified that this project report entitled "**FRED**" is a bonafide work of **Keshav (18BCE1041), Saurabh Mohata (18BCE1207), Aditya Kumar (18BCE1330)** and they carried out the Project work under my supervision and guidance for CSE3999 - Technical Answers for Real World Problems (TARP).

Dr. Malathi G.
SCOPE, VIT Chennai

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Malathi G.**, School of Computer Science and Engineering for her consistent encouragement and valuable guidance offered to us throughout the course of the project work.

We are extremely grateful to **Dr. R. Jagadeesh Kannan**, Dean and **Dr. S. Geetha**, Associate Dean, School of Computer Science and Engineering (SCOPE), Vellore Institute of Technology, Chennai, for extending the facilities of the School towards our project and for their unstinting support.

We express our thanks to **Dr. Justus S,** Head of the Department, B.Tech. Computer Science and Engineering for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the courses.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

KESHAV	Mohata	
Keshav **18BCE1041**	**Saurabh Mohata** **18BCE1207**	**Aditya Kumar** **18BCE1330**

ABSTRACT

The major problem faced by young students is to find good high-quality educational content. Often high-quality free content gets weeded by other mediocre quality paid courses because of their aggressive marketing strategies. Obviously, it is not possible for a non-profit entity to market as aggressively as their profit-making counterparts.

We are building a website where you can search for a topic you want to learn and it will suggest you the best free resources available for the topic. There will be a feature for sorting the content based on different parameters including the type of content (video, blog, course) and time required for its completion. It will provide you links to best available resources for that topic. The most challenging part is to surface out good content creators from average creators who are already very famous.

At FRED, our aim is to highlight free first-rate educational content providers. We want to bridge the gap between entities that provide free high-quality educational content and the people who have a desire to learn and improve but aren't able to do so because of various different reasons viz. unaffordable fees, are unaware of free good content, are paralyzed by the number of choices that are available, etc.

CONTENTS

1. Introduction

1.1 Objective and goal of the project

The objective of this project is to build a website which will be able to recommend the user's free resource material required by them inorder to study for a specific domain. This website is built for the users who find it tough to get good quality resources, and often struggle to get them free of any cost. Many of times these users ask other people regarding good quality educational resources so we decided to form a website which will help the users in doing so.

At FRED, our aim is to highlight free first-rate educational content providers. We want to bridge the gap between entities that provide free high-quality educational content and the people who have a desire to learn and improve but aren't able to do so because of various different reasons viz. unaffordable fees, are unaware of free good content, are paralyzed by the number of choices that are available, etc.

We wish to build a website which can solve all these problems while making it very user friendly. On FRED, each user will be able to recommend good quality free resources and the other fellow users can see the recommended course. This will also help in building a strong user community which is close knitted.

1.2 Problem Statement

The major problem faced by young students is to find good high-quality educational content. Often high-quality free content gets weeded by other mediocre quality paid courses because of their aggressive marketing strategies. Obviously, it is not possible for a non-profit entity to market as aggressively as their profit-making counterparts.

We are building a website where you can search for a topic you want to learn and it will suggest you the best free resources available for the topic.

These resources will be categorized in 2 parts, Approved Courses and Not Approved courses. Not Approved will list the courses which are not preferred by many users whereas Approved means the courses which are preferred by the a number of users.

FRED aims to build a strong user community as these users will have a functional role to play inorder for the website to work.

Each user will have a Reputation which will be increased by other fellow users while visiting his/her profile.

Any user will be able to recommend content by filling a very simple form and whenever a new content item is added, for example, a course, a blog or a video, it goes into the Unapproved list. The other's user will have an option if they want to add their reputation to the content item's reputation by choosing the 'Recommend This!' option. And if the content item's reputation reaches the threshold of 5000, then that item will move to the Approved list.

We also can view these recommended courses on our website. While viewing the list, each user will be given the option to either Upvote or Downvote the item and these items will be sorted based on the Ratio of Upvotes and Downvotes. There will be a feature for sorting the content based on different parameters including the type of content (video, blog, and course). It will provide users links to best available resources for that topic. The most challenging part is to surface out good content creators from average creators who are already very famous.

1.3 Motivation

The motivation behind building such a website is quite simple and straightforward. We as students whenever want to learn about a domain, for example, if you are a CSE student who wants to learn Web Development, you will look on the internet which is easily accessible, about courses or blogs or educational videos about and most probably choose a course. The majority of the websites for example Udemy, Coursera will have a number of courses related to Web Development which will be either free or paid courses. Now due to such vast availability of courses, it will become confusing for us to choose which course to do. Also these kind of websites will push paid courses which are expensive, more to its users inorder to generate funds for themselves. These websites will not market the free courses, the way they market the paid ones.

So the main motive behind FRED is to solve all of the above issues. FRED will list free courses, blogs and videos which will be recommended by fellow users of the website. We wish to build a user community that will help us in the functioning of the website by using concepts such as reputation and the more the number of users, the more the reputation of a user, thus making it easier for the recommendation of any type of content.

1.4 Challenges

There are various challenges that the team has to overcome inorder to ensure the smooth working of the complete project as a system. The major challenge is to build the system using the MERN stack, which involves the concepts of MongoDB, ExpressJS, ReactJS and NodeJS. Each of these needs to be integrated into different modules of the system which will be a huge challenge. Also, the connectivity of all of these concepts needs to be perfectly done as only then we can ensure the smooth working of the project.

Inorder for the project to function as we want it to, we need to work on 2 major things, make the UI really user-friendly, while making it attractive and secondly incorporating some complex functionalities inorder for the website to work.
We need to make the motive of using our platform, FRED, clear to the final user, i.e. the people who are looking for resources to learn about a specific domain free of cost. The resources can be provided in the form of a blog, video or a course link.

2. Literature Survey

S.No.	Title	Conference/ Journal	Authors	Dataset Used	Methodology Used	Performance	Conclusion	Future Scope
1	E-Learning during Lockdown of Covid-19 Pandemic: A Global Perspective	International Journal of Control and Automation Vol. 13, no. 4, pp. 1088-1099, Jun. 2020.	R.Radha, K.Mahalakshmi, Dr.V.Sathish Kumar, Dr.AR.Saravanakumar	Primary data collected from undergraduate students	Online questionnaire was used to get a detailed survey	Among 175 respondents, around 82.29 % of students reported their willingness to learn from e-sources. Around 82.86 % of students have reported their self-study skills to improve because of e-learning. 80.57 % of students are opinioned that e-learning is very useful during the quarantine time. 70.86 % are esteemed that e-learning makes their knowledge extensive.	Majority of the students showed willingness towards E-learning. They were also satisfied with the current trend of online learning and also reported that their self-study skills have improved. They were keen on utilizing their quarantine time by learning on new skills online.	E-learning seems to be the forthcoming trend. In the future, this e-learning module made unavoidable option but this paper does not address the side of people who do not have access to proper source of e-learning.
2	A study of students' perception about e-learning	Indian Journal of Clinical Anatomy and Physiology, October-December, 2018;5(4): 501-507	Mehandi Vinayak Mahajan, Kalpana. R	Primary data collected from students (2nd year M.B.BS) of age group 19-21yrs of different socioeconomic and demographic backgrounds.	Structured questionnaire	98% agree to find e-learning useful and are motivated to use it. 75% believe that they will have ready access to e-learning courses while other don't due to lack of constant supply of internet. 38% agree e-learning is disadvantages as it will replace faculties as they are comfortable with the traditional teaching style, while 26% disagree. Students gave suggestions to start blogs, online discussions, online submission of home work, and video assisted training for clinical work.	The study clearly depicts that e-learning has its benefits from a student's perspective and it will have a positive influence on their performance with better understanding of their courses. Thus for at least the following few years, the university needs come out with e-learning tools and modules for a better teaching -learning experience make a positive impact on the students career.	E-learning still has a lot of hurdles to cross inorder to penetrate into deepest of parts of society.
3	Online	Journal of	Shivangi	This study is	The research	EdTech Start-ups are	E-learning can	Online

	Learning: A Panacea in the Time of COVID-19 Crisis	Educational Technology Systems 2020, Vol. 49(1) 5–22	Dhawan	completely based on the secondary data. Secondary sources of data used are (a) journals, (b) reports, (c) search engines, (d) company websites and scholarly articles, (e) research papers, and other academic publications.	tool used for analyzing the data which amassed from different sources for this study is a content analysis and the research method is descriptive research. We have taken into consideration the qualitative aspects of the research study.	tapping all the right opportunities by providing free online courses to students amidst this crisis. UNESCO also suggested that these EdTech Start-ups and learning apps can help students during such hard times. According to the reports by KPMG and Google, the EdTech sector will boom and is likely to reach around 2 Billion Dollars by 2021.	help in providing inclusive education even at the time of crisis. Such systems need to be developed in educational institutions that make sure that no student is getting deprived of education due to their location, social class, ethnicity, and so on.	methods of teaching support and facilitate learning–teaching activities, but there is a dire need to weigh the pros and cons of technology and harness its potentials
4	A Comparative Analysis of MOOC (Massive Open Online Course) Platforms	Informatica Economică vol. 20, no. 2/2016	Maria Conache, Ramona Dima, Andreea Mutu	Qualitative and Quantitative data. The quantitative approach is based on comparing web traffic data and specific measurements for each platform. The qualitative approach is about describing the characteristics of each platform according to specific criteria.	In order to present comparatively the features of four of the most popular MOOC platforms: Coursera, Udemy, Udacity and EdX , authors have focused on the following measurements and criteria: 1. by the users criteria: - number of visitors - student's profile 2. by the interest of each platform to gain more popularity: - time spent by a user on the website - ratings on mobile apps - number of partners - number of	Coursera, Udacity and Udemy are for-profit organizations. EdX is non-profit.y Coursera and EdX provide mostly university-style courses because of their partnership with elite universities around the world. Udemy hosts courses from individual instructors and collaborates with affiliates that promote the courses. All four providers offer both free and paid courses. Some of them allow students to access all course content without paying fees while others limit features like verified certificates or assessments. Udemy hosts more than 40000 courses, next is Coursera with 1872 courses, followed by EdX with 913 courses and Udacity with 131 courses at the time of this study.	All four platforms that can be chosen for online learning and are very popular among people of every continent. It seems that Coursera and Udemy are more visited and, as global ranking, they are in the top 10 of the Category Rank for Career and Education. It's noticeable that Udemy is the most visited website, but is not the best ranked and Udacity has the longest time spent on site by a user.	With the number of courses offered being so large it is really difficult to find sustainable and good courses which are of great help. So that issue needs to addressed in future.

					courses - foreign languages support			
5	The Comparison of MOOC (Massive Open Online Course) Platforms of edX and Coursera (study case: student of programming courses)	2018 International Conference on Information Management and Technology (ICIMTech), Jakarta, Indonesia, 2018, pp. 339-344.	Tanty Oktavia, Harjanto Prabowo, Meyliana, Suhono Harso Supangkat	The data source of this study is the experiment using two MOOC platforms, which are edX and Coursera	All participants in this experiment are filtered by some of the criteria, such as they already used MOOC platform before either edX or Coursera, and they take the similar learning course. For this study, participants take programming course for one period of registration. Then for results authors use t-test for 2 group samples. The result will show the significant features of MOOCs to support learning activities.	Based on the RASE model and the concept of e-learning activity (Do, Absorb, Connect), it can be concluded that those features contained in edX and Coursera have similar functions and activities of the learner's perspectives. However, for the support component, the function of the video service in Coursera shows that the satisfaction rate is still below the average, reaching 2.23. While in the characteristics of edX, bookmark and tracking features also show a number below the average that is 2.4.	EdX ranks better in terms of video service but Coursera has a higher ranking for user satisfaction as compared to EdX. We get to learn that based on the criteria and the user's preference how we can develop a platform by combing these integral features of platforms.	This paper only focuses on a limited criterion for distinguishing between the two platforms. A much extensive differentiation can be made if we do a much detail data analysis.
6	What Do Current College Students Think about MOOCs?	MERLOT Journal of Online Learning and Teaching Vol. 11, No. 2, June 2015	Andrew W. Cole, C. Erik Timmerman	Data was collected from a sample of college students, the primary audience ultimately targeted by many current developers of MOOC	The researchers advertised the study in several undergraduate courses. Students who expressed interest in participating were able to click a link in	The thematic category most prevalent in the data dealt with the range of issues that students raised about the reliability of MOOCs as a form of instruction. Such issues were addressed by 81% of participants. Many students view MOOCs positively because they make	The open nature of MOOCs is seen by many students as bringing the advantages of higher education to a population that previously may have been unable to access higher education. MOOC benefit is that anybody with valuable information can share that information with	Students and their responses need to be anayzed for a much extensive review and should focus on new criteria for evolving culture of e-

				offerings.	a solicitation email, which took them to a web-based survey containing open-ended questions about students’ perceptions of MOOCs. Then this data was analysed using thematic analysis.	education and knowledge more accessible to more people	others. However, unlike in a traditional college course, that material is viewed as potentially unreliable, so the course itself can be viewed as potentially unreliable as well.	learning integrating it with concepts of MOOCs.

3. Requirements Specification

3.1 Hardware Requirements

A computer with the standard processor and decent RAM which can handle multiple applications at a certain time.

3.2 Software Requirements

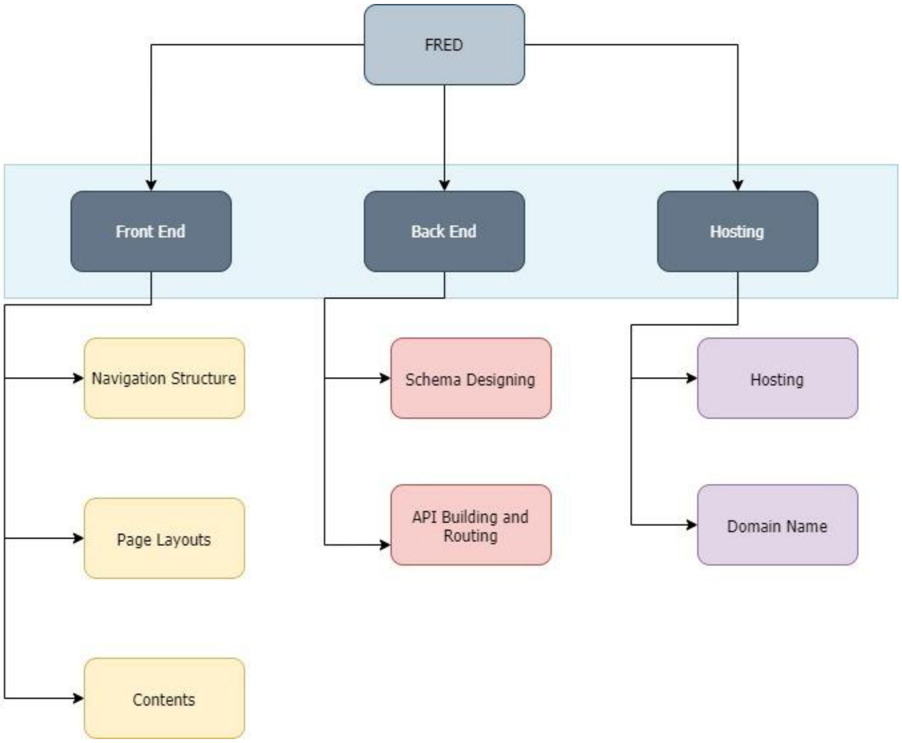
For development of the website we need any Operating System having the following applications –

- Any browser
- Visual Studio Code
- Notepad++
- Codesandbox.io
- Pesticide Extension On Google Chrome
- NPM
- Filesync

For visiting the website a user will only need a browser.

4. System Design

Architecture Diagram



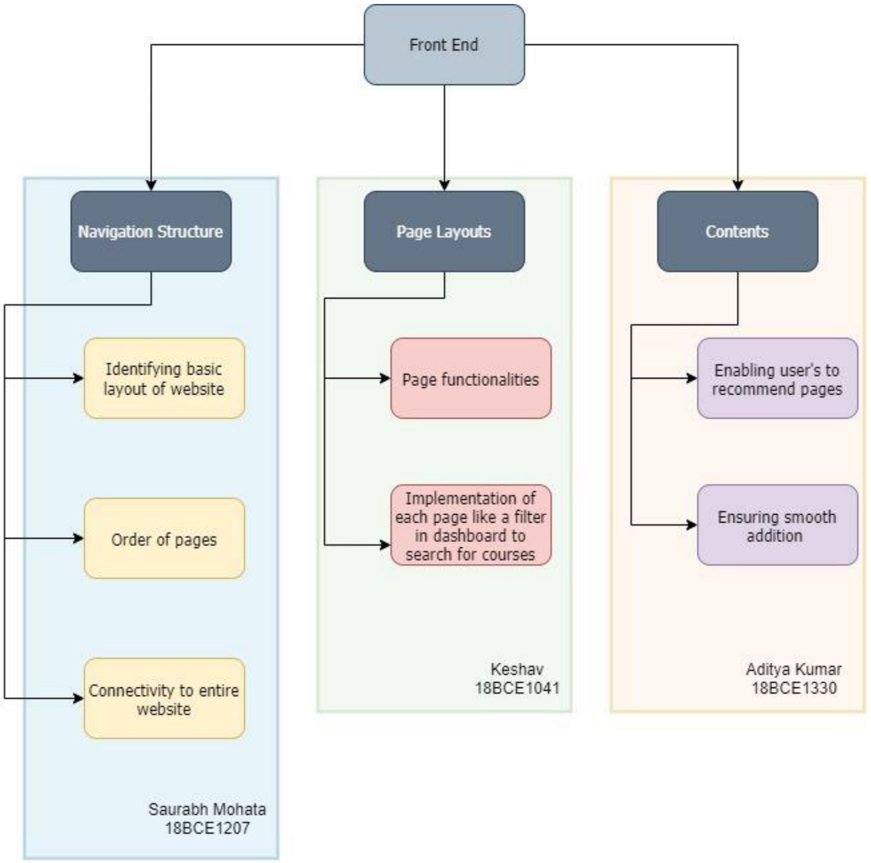
We have divided the development of the system into mainly parts –

1. Front-End
2. Back-End
3. Hosting

Each of these parts are further explained below and along with the breakdown of each part into small modules.

1. Front-End

This part of the project is responsible to the look and feel of the project and handles how the user will interact with the website. This module needs to be user-friendly and simple to use while keeping intact the complex functionalities of the website.


Front End
Navigation Structure
Page Layouts
Contents
Identifying basic layout of website
Order of pages
Connectivity to entire website
Saurabh Mohata 18BCE1207
Page functionalities
Implementation of each page like a filter in dashboard to search for courses
Keshav 18BCE1041
Enabling user's to recommend pages
Ensuring smooth addition
Aditya Kumar 18BCE1330

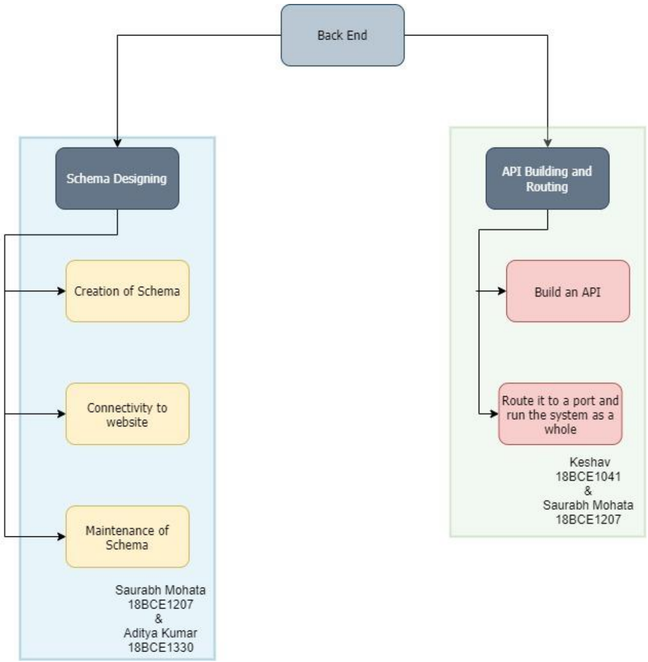
1.1. Navigation structure - The navigation structure is the order of the pages, the collection of what links to what. Navigation will be an important feature of our site and we are planning to develop an intuitive and easy-to-navigate website by avoiding excessive clicks or page links which will make it fast and easy for the users to visit other sections of the website.

1.2. Page Layouts - As our site will have a lot of links to browse from and so it is essential to have a good page layout which allows the user to find quickly what they seek.

1.3. Contents - After initial contents being set up from us, we want the users to update the further contents in the website without having us to directly edit the code. So, we need a robust updation system which allows the user to update the content easily on a regular basis.

2. Back-End

This part of the website will act as the backbone of the complete system and will handle the user's request. Each of user's action on the front-end needs to be mapped with an action on the functioning of website, this module will help on that. This module includes schema/database design and updation along with the making of connections of the back-end to the front-end functions.



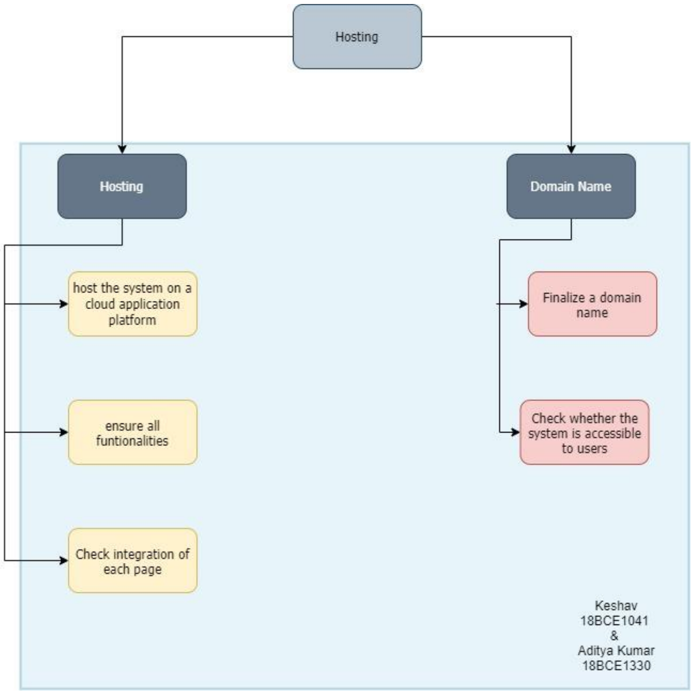
2.1. Schema Designing - Databases allow us to store, sort, search through, and display large amounts of information. This is a very crucial feature for our site which will store all the links based on various types of the content.

2.2. API building and Routing - API stands for application programming interface. It is a set of definitions and protocols for building and integrating application software.

We will be using it to have the communications between different services of the application with proper routing such that the logic used is not being exposed to the client side

3. Hosting

This module is divided into 2 modules –

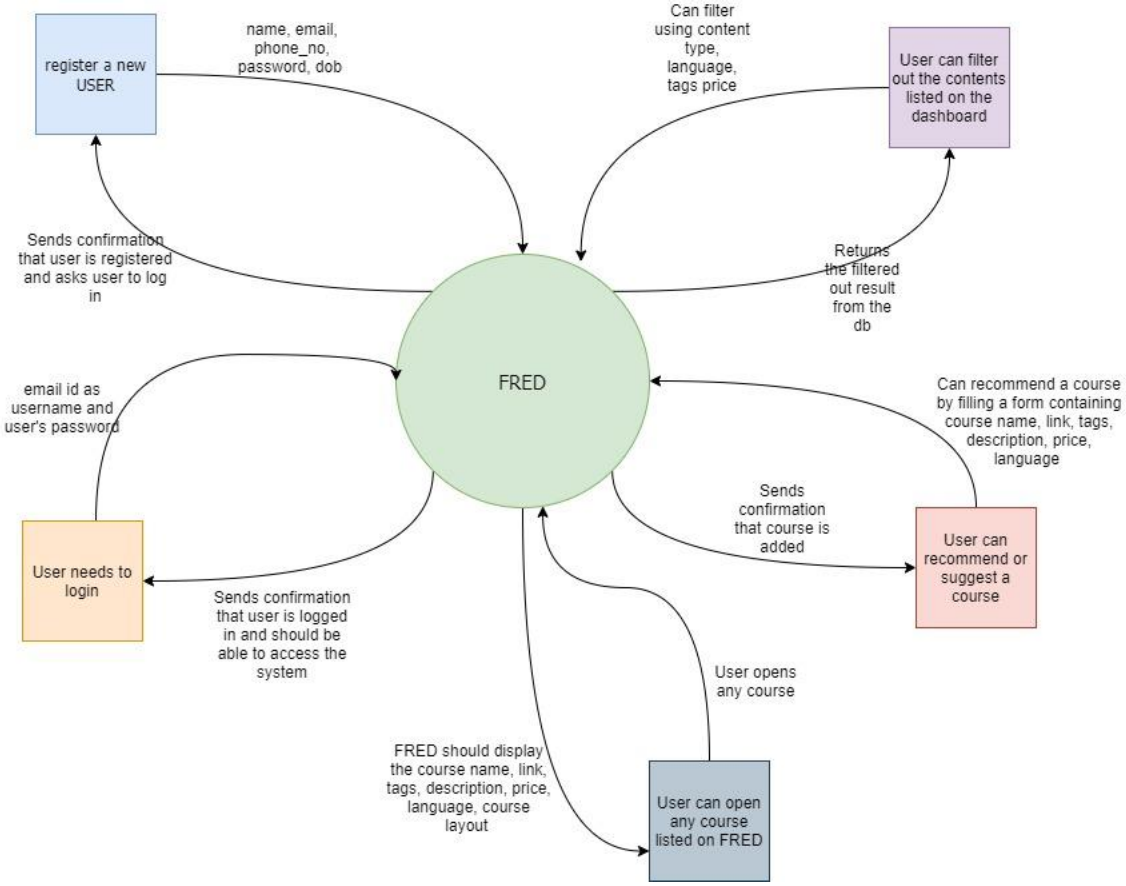

Hosting
Hosting
Domain Name
host the system on a cloud application platform
ensure all funtionalities
Check integration of each page
Finalize a domain name
Check whether the system is accessible to users
Keshav
18BCE1041
&
Aditya Kumar
18BCE1330

3.1. Hosting – This module includes ensuring the working of entire system as a whole. It should check all the frond-end to back-end connections. It involves testing the website thoroughly and checking if each functionality is working or not.

3.2. Domain name - It is where our website will physically be located. On a server, somewhere, are a set of files that are transmitted to user computers when the user search for our domain name which will be the address of our site.

5. Implementation of System

Context Diagram


register a new USER
name, email, phone_no, password, dob
Sends confirmation that user is registered and asks user to log in
Can filter using content type, language, tags price
User can filter out the contents listed on the dashboard
Returns the filtered out result from the db
FRED
email id as username and user's password
User needs to login
Sends confirmation that user is logged in and should be able to access the system
Can recommend a course by filling a form containing course name, link, tags, description, price, language
Sends confirmation that course is added
User can recommend or suggest a course
User opens any course
FRED should display the course name, link, tags, description, price, language, course layout
User can open any course listed on FRED

Basic **ideology** of the system -

Each user will have a Reputation which will be increased by other fellow users while visiting the profile.

Any user will be able to recommend content by filling a very simple form and whenever a new content item is added, for example, a course, a blog or a video, it goes into the **Unapproved list**. The other's user will have an option if they want to add their reputation to the content item's reputation by choosing the 'Recommend This!' option. And if the content item's reputation reaches the threshold of 5000, then that item will move to the **Approved list**.

While viewing the list, each user will be given the option to either Upvote or Downvote the item and these items will be sorted based on the Ratio of Upvotes and Downvotes.

Implementation –

For creating this website, we first created basic static pages using codesandbox.io. This is a very useful tool providing us with the ability to make interactive webpages while giving us a layout of the page at all times.

While the front-end static pages were being made, at the same time back-end was worked upon too, and APIs for all of the functions were created. APIs were then thoroughly tested one by one and we mostly used postman test method. After the success of tests, we then integrated the APIs to the static pages. Then each static page was tested again along with the specific API.

Once all of the pages were merged with their APIs, then the system was merged as a whole by combining all the static pages along with their APIs.

Once Front-End and Back-End were connected, thorough tests were conducted by checking each and every functionality of the website and if any errors were encountered, they were resolved immediately.

6. Results and Discussion

We have built a website, FRED – Free Education, and as the name suggests, we have been successful in providing a portal/website which will be of great use to the end users, i.e. the students who are willing to learn by providing them with great content which is actually free of cost. FRED is unique website as it lists educational courses, blogs and videos in whichever field/domain the user wants to learn about. We have not restricted the website to a certain category like Udemy or Coursera which list only Courses, similarly not YouTube which is very vast and has only videos.

We have built the website using MERN stack, which includes MongoDB, Express JS, React JS and Node JS. The integration of all these modules was very tough but we have successfully managed to do so. The project is fully working with all the desired outcomes as listed in the proposed system.

The database/schema related modules which were handled by Saurabh Mohata, are working on real-time basis and are being updated accordingly if any changes are made via the front-end like the process to shift a course from the unapproved section to the approved section or the process of keeping a steady log of the user's reputation and also creating a verification API whenever a new user signs up.

The front-end pages which were built by Aditya Kumar and Keshav are very user friendly, attractive in nature and offer a great UI experience. They are relatively simple to use in order to integrate the user-friendly nature which the website offers while including the complex functionalities such as reputation of a course and the user's reputation or the course listing using Upvotes and Downvotes and the actual recommendation process of the website.

The connections between the Front-end and the Back-end were made by Keshav who was also responsible for the process of complete deployment of the project as a whole. He has managed to integrate both, the front-end and the back-end perfectly and everything is in great sync.

7. Conclusion and Future Work

In due course of time, while we were completing this project, we had to face various obstacles but we treated them as opportunities to learn various forms of new things. We followed the trial and error ideology, we learnt from our mistakes, corrected them and kept on working hard until we were able to get the full project fully functioning. The project was a great challenge as it was built on MERN stack technology, which includes concepts of MongoDB, Express JS, React JS and Node JS which was tough to integrate in each of the modules and we also realised as how small changes can make big impact on the whole system. This project has certainly helped us in giving the basic idea of what it takes to become a professional who works in Web Development environment. For successful completion of this project, we needed a very thorough and deep understanding of the concepts and for telling us the right practical implementations of the concepts, we would really like to take this opportunity to thank Dr. Malathi G. ma'am, our faculty for the course TARP, Technical Answers for Real Time Problems who helped us with her valuable inputs in times of need.

In our system, we can see that we have successfully managed to build a whole system which satisfies the requirements mentioned by us in the proposed system section. Our system is fully working and has no error as we have tested it thoroughly, be it the functionality of real-time database updation and also the smooth and user friendly UI which will help us in attracting users. We have created a website which will be useful for students who are interested to learn in any of the domains, like be it CSE related Domains or any other subject or field related. FRED – FREE EDUCATION, as the name suggests, the website is of great use as it is built while keeping the final users, i.e. the students in mind, by providing them resources or access to great content which is actually free and not in order to promote or market the website to generate funds.

The project can be hosted on a domain, we can buy a domain and make it available for general public to use.

For the future, we can enhance the project using following ways, The whole project will actually be successful if we gain or attract more number of users, so we need to gain attraction and draw user’s attention to actually use our website. As more number of user’s are present on the website, it will be very much helpful for the process of approval of courses as each course is being approved based on addition of user’s reputation to a certain threshold set by us, 5000. So as we have more users who are willing to recommend and add/associate/contribute their reputation to a course’s reputation the course will then be moved from the unapproved courses list to the approved courses list. Also, if we will have more users, each user can add to another user’s reputation on the website.

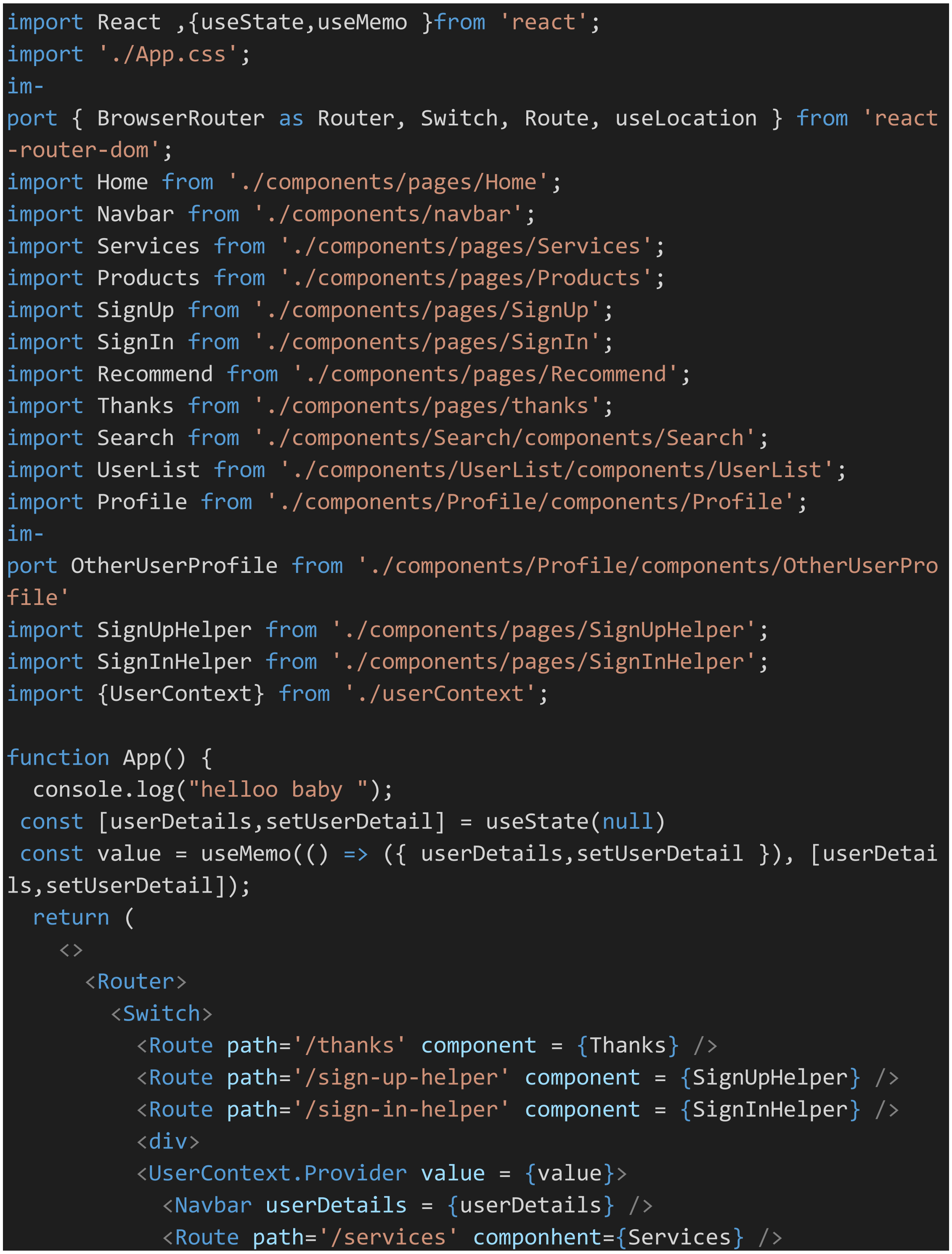
We can have a functionality of sorting the courses in a certain way, for example, in our current project we are sorting the courses on the basis of the **ratio** of Upvotes and Downvotes, but we can create a sorting function which will sort the listed courses based on just the number of Upvotes or Downvotes.

8. REFERENCES

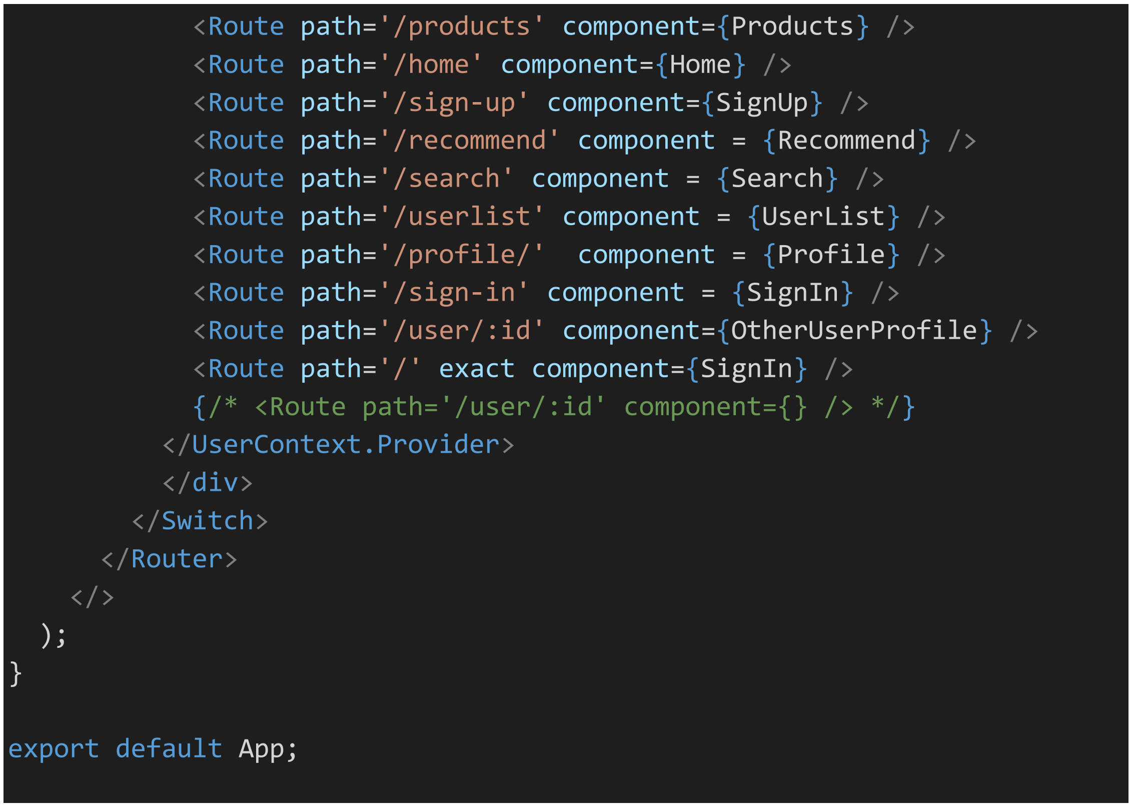
- R.Radha, K.Mahalakshmi, Dr.V.Sathish Kumar, Dr.AR.Saravanakumar. (2020). E-Learning during Lockdown of Covid-19 Pandemic: A Global Perspective. International Journal of Control and Automation, 13(4), 1088-1099. Retrieved from http://sersc.org/journals/index.php/IJCA/article/view/26035
- Mahajan M V, Kalpana R, A study of students' perception about e-learning. Indian J Clin Anat Physiol 2018;5(4):501-507
- Dhawan S. Online Learning: A Panacea in the Time of COVID-19 Crisis. Journal of Educational Technology Systems. 2020;49(1):5-22. doi:10.1177/0047239520934018
- Maria CONACHE & Ramona DIMA & Andreea MUTU, 2016. "A Comparative Analysis of MOOC (Massive Open Online Course) Platforms," Informatica Economica, Academy of Economic Studies - Bucharest, Romania, vol. 20(2), pages 4-14.
- T. Oktavia, H. Prabowo, Meyliana and S. H. Supangkat, "The Comparison of MOOC (Massive Open Online Course) Platforms of edX and Coursera (Study Case: Student of Programming Courses)," 2018 International Conference on Information Management and Technology (ICIMTech), 2018, pp. 339-344, doi: 10.1109/ICIMTech.2018.8528178.
- Cole, A.W. & Timmerman, C.E.. (2015). What do current college students think about MOOCs?. 11. 188-201.
- https://www.w3schools.com/whatis/
- https://reactjs.org/docs/create-a-new-react-app.html
- https://www.mongodb.com/mern-stack

APPENDIX

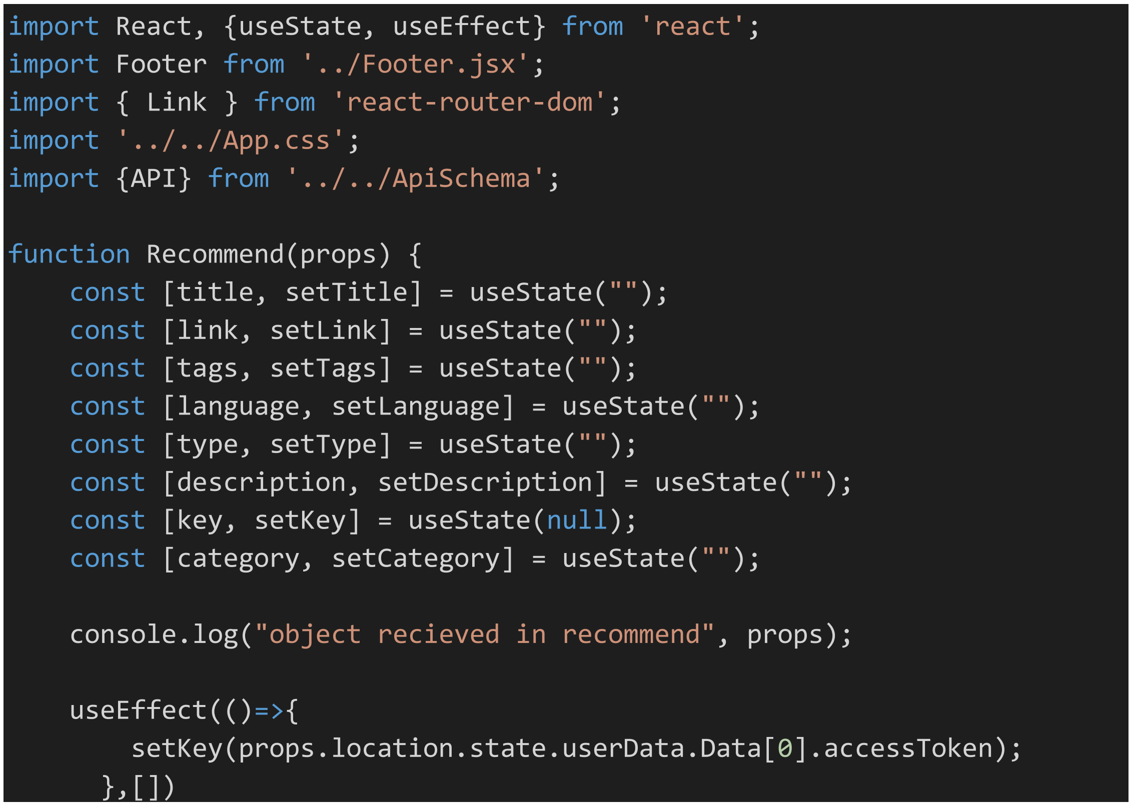
SAMPLE FRONTEND CODE:



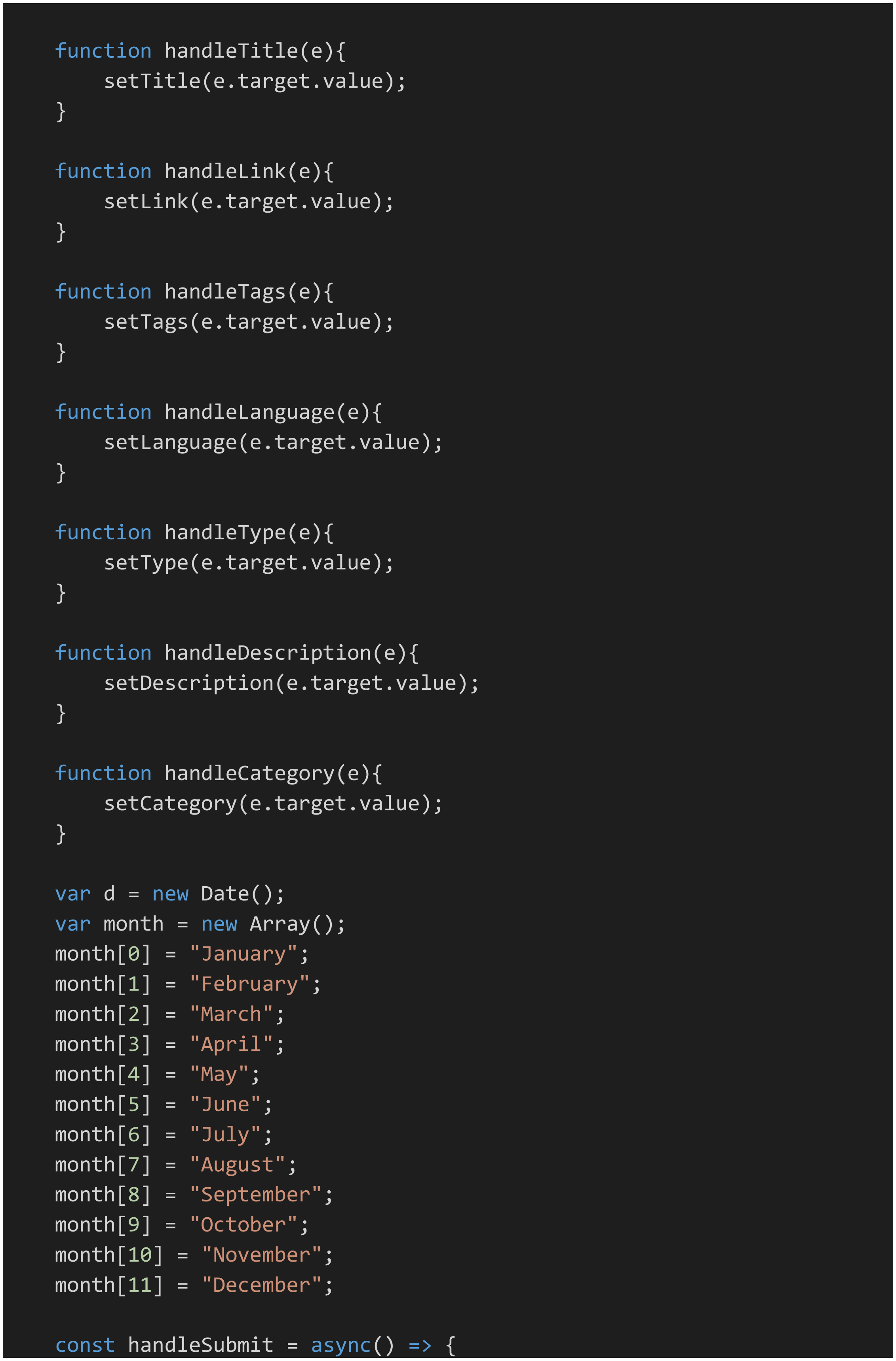
```
import React ,{useState,useMemo }from 'react';
import './App.css';
import { BrowserRouter as Router, Switch, Route, useLocation } from 'react-router-dom';
import Home from './components/pages/Home';
import Navbar from './components/navbar';
import Services from './components/pages/Services';
import Products from './components/pages/Products';
import SignUp from './components/pages/SignUp';
import SignIn from './components/pages/SignIn';
import Recommend from './components/pages/Recommend';
import Thanks from './components/pages/thanks';
import Search from './components/Search/components/Search';
import UserList from './components/UserList/components/UserList';
import Profile from './components/Profile/components/Profile';
import OtherUserProfile from './components/Profile/components/OtherUserProfile'
import SignUpHelper from './components/pages/SignUpHelper';
import SignInHelper from './components/pages/SignInHelper';
import {UserContext} from './userContext';

function App() {
  console.log("helloo baby ");
 const [userDetails,setUserDetail] = useState(null)
 const value = useMemo(() => ({ userDetails,setUserDetail }), [userDetails,setUserDetail]);
  return (
    <>
      <Router>
        <Switch>
          <Route path='/thanks' component = {Thanks} />
          <Route path='/sign-up-helper' component = {SignUpHelper} />
          <Route path='/sign-in-helper' component = {SignInHelper} />
          <div>
          <UserContext.Provider value = {value}>
            <Navbar userDetails = {userDetails} />
            <Route path='/services' componhent={Services} />
```



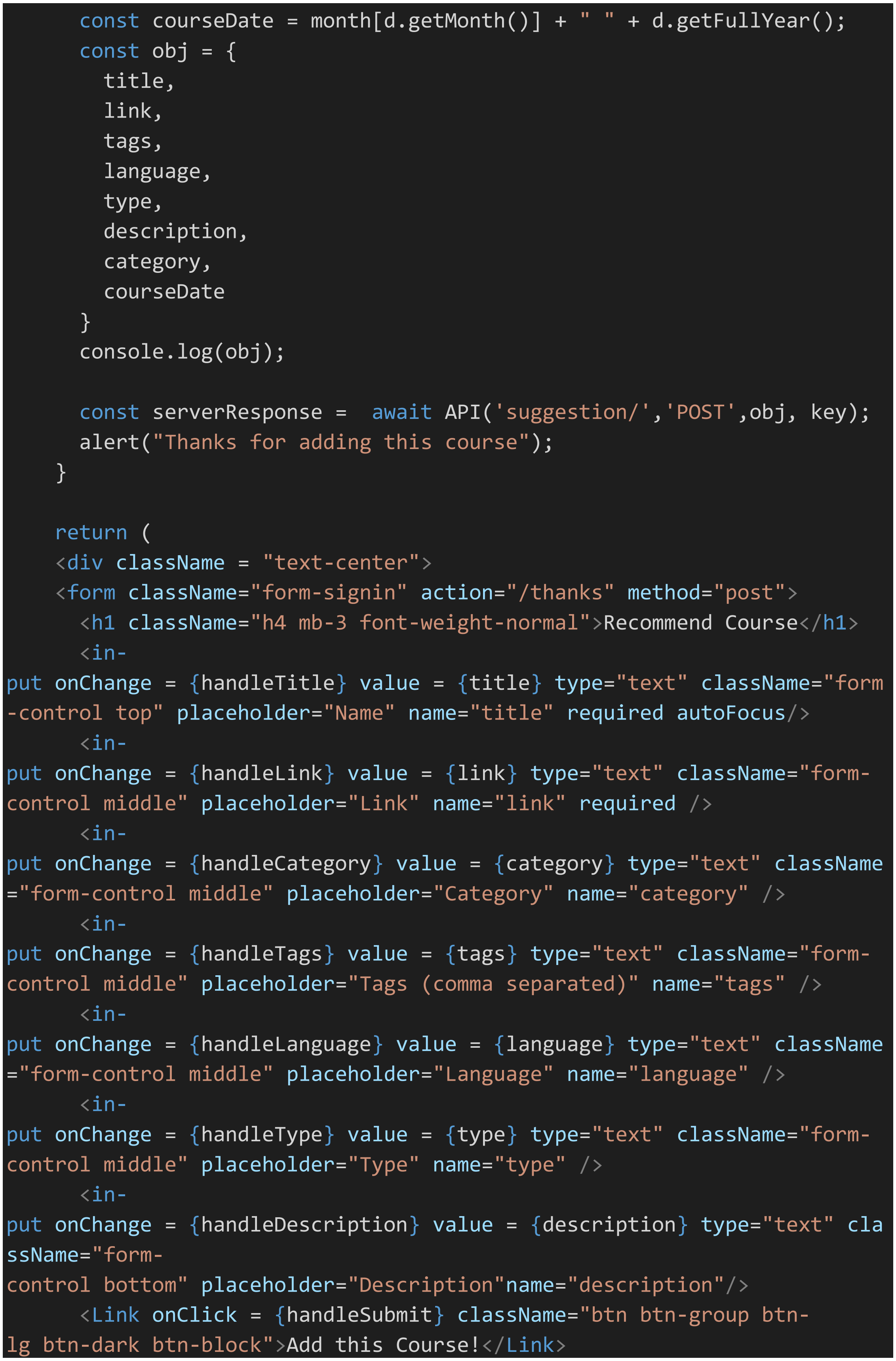
```
            <Route path='/products' component={Products} />
            <Route path='/home' component={Home} />
            <Route path='/sign-up' component={SignUp} />
            <Route path='/recommend' component = {Recommend} />
            <Route path='/search' component = {Search} />
            <Route path='/userlist' component = {UserList} />
            <Route path='/profile/'  component = {Profile} />
            <Route path='/sign-in' component = {SignIn} />
            <Route path='/user/:id' component={OtherUserProfile} />
            <Route path='/' exact component={SignIn} />
            {/* <Route path='/user/:id' component={} /> */}
          </UserContext.Provider>
          </div>
        </Switch>
      </Router>
    </>
  );
}

export default App;
```



```
import React, {useState, useEffect} from 'react';
import Footer from '../Footer.jsx';
import { Link } from 'react-router-dom';
import '../../App.css';
import {API} from '../../ApiSchema';

function Recommend(props) {
    const [title, setTitle] = useState("");
    const [link, setLink] = useState("");
    const [tags, setTags] = useState("");
    const [language, setLanguage] = useState("");
    const [type, setType] = useState("");
    const [description, setDescription] = useState("");
    const [key, setKey] = useState(null);
    const [category, setCategory] = useState("");

    console.log("object recieved in recommend", props);

    useEffect(()=>{
        setKey(props.location.state.userData.Data[0].accessToken);
      },[])
```



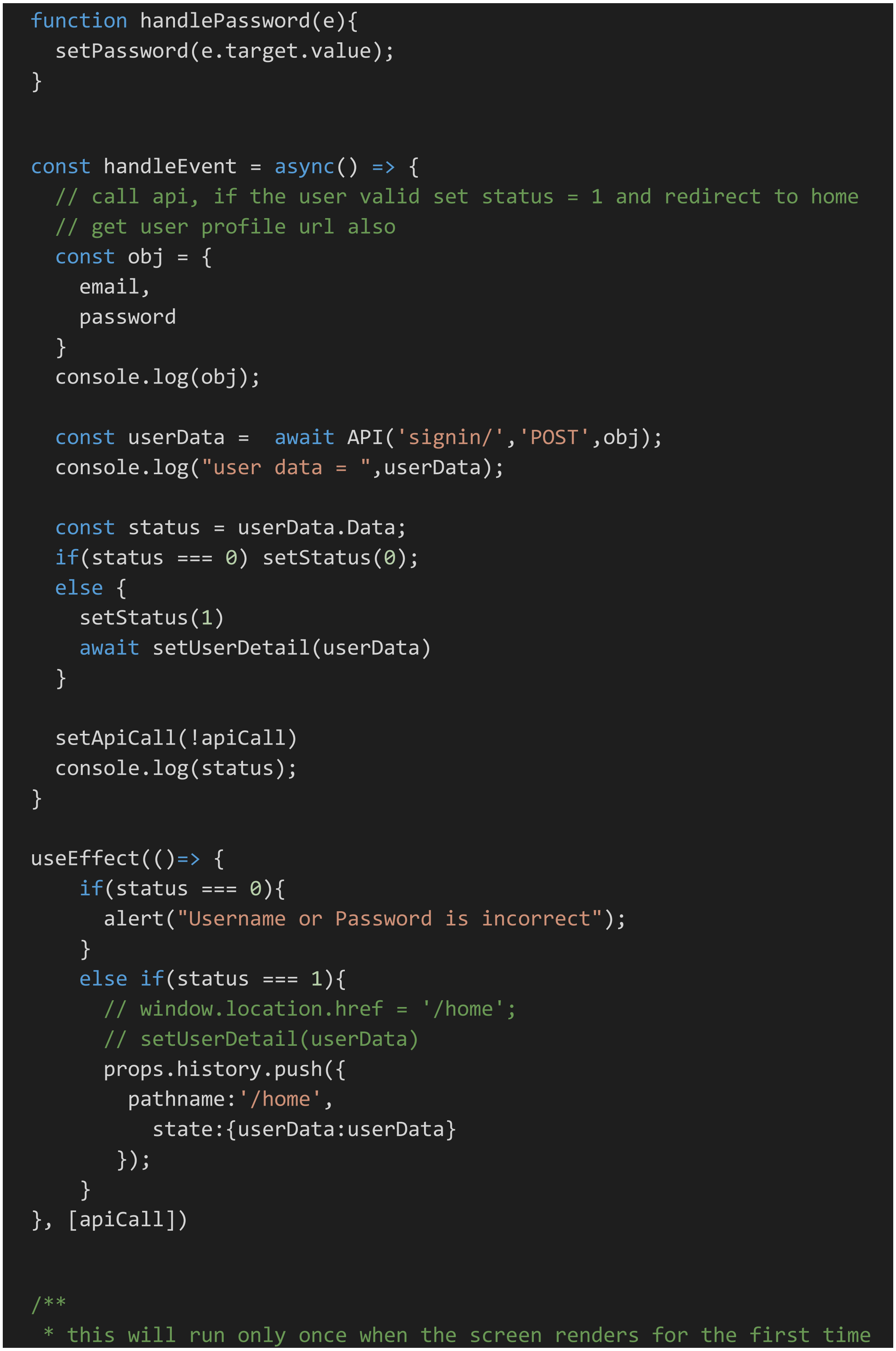
```
function handleTitle(e){
    setTitle(e.target.value);
}

function handleLink(e){
    setLink(e.target.value);
}

function handleTags(e){
    setTags(e.target.value);
}

function handleLanguage(e){
    setLanguage(e.target.value);
}

function handleType(e){
    setType(e.target.value);
}

function handleDescription(e){
    setDescription(e.target.value);
}

function handleCategory(e){
    setCategory(e.target.value);
}

var d = new Date();
var month = new Array();
month[0] = "January";
month[1] = "February";
month[2] = "March";
month[3] = "April";
month[4] = "May";
month[5] = "June";
month[6] = "July";
month[7] = "August";
month[8] = "September";
month[9] = "October";
month[10] = "November";
month[11] = "December";

const handleSubmit = async() => {
```



```
   const courseDate = month[d.getMonth()] + " " + d.getFullYear();
   const obj = {
     title,
     link,
     tags,
     language,
     type,
     description,
     category,
     courseDate
   }
   console.log(obj);

   const serverResponse =  await API('suggestion/','POST',obj, key);
   alert("Thanks for adding this course");
  }

  return (
  <div className = "text-center">
  <form className="form-signin" action="/thanks" method="post">
   <h1 className="h4 mb-3 font-weight-normal">Recommend Course</h1>
   <in-
put onChange = {handleTitle} value = {title} type="text" className="form
-control top" placeholder="Name" name="title" required autoFocus/>
   <in-
put onChange = {handleLink} value = {link} type="text" className="form-
control middle" placeholder="Link" name="link" required />
   <in-
put onChange = {handleCategory} value = {category} type="text" className
="form-control middle" placeholder="Category" name="category" />
   <in-
put onChange = {handleTags} value = {tags} type="text" className="form-
control middle" placeholder="Tags (comma separated)" name="tags" />
   <in-
put onChange = {handleLanguage} value = {language} type="text" className
="form-control middle" placeholder="Language" name="language" />
   <in-
put onChange = {handleType} value = {type} type="text" className="form-
control middle" placeholder="Type" name="type" />
   <in-
put onChange = {handleDescription} value = {description} type="text" cla
ssName="form-
control bottom" placeholder="Description"name="description"/>
   <Link onClick = {handleSubmit} className="btn btn-group btn-
lg btn-dark btn-block">Add this Course!</Link>
```

```
      <p className="mt-5 mb-3 text-muted">&copy; Fred</p>
    </form>
    <Footer />
    </div>
    )
}

export default Recommend;
```

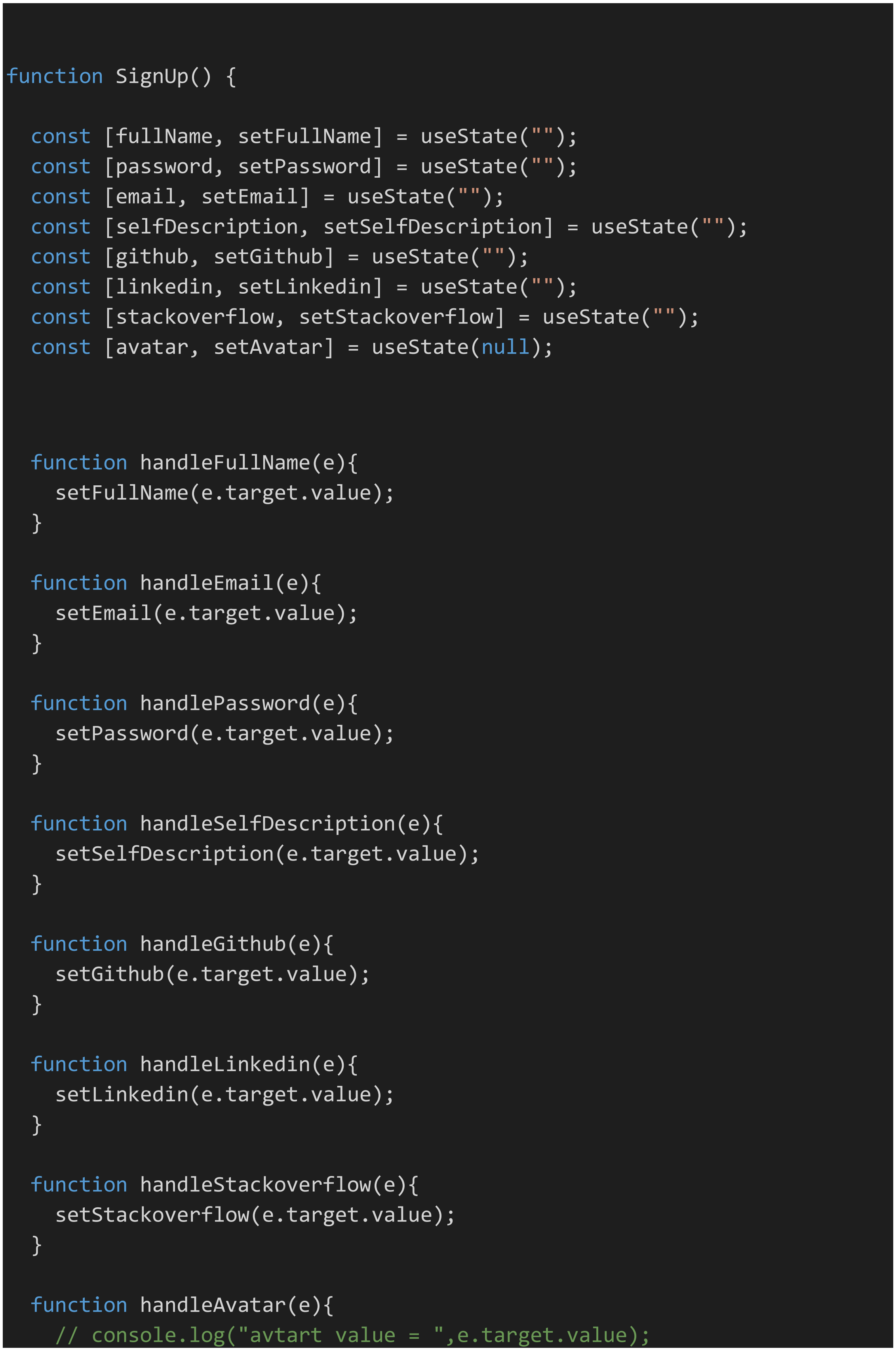
```
import React, {useState, useEffect, useContext } from 'react';
import Footer from '../Footer.jsx';
import { Link ,Redirect} from 'react-router-dom';
import '../../App.css';
import {UserContext} from '../../userContext';


const API = async(url,method,obj = {}) =>{
  const res = await fetch('http://localhost:9000/webApi/'+ url,{
    method:method,
    headers: {
      'Content-Type': 'application/json'
     },
    body:JSON.stringify(obj)
  })
  const data = await res.json();
  // console.log("data = ",data);
  return data;
}

function SignIn(props) {
  const {userDetails,setUserDetail }= useContext(UserContext);
  // console.log("hello baby again" , userDetails);
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [userData, setUserData] = useState();

  const [status, setStatus] = useState(2);
  const [apiCall,setApiCall] = useState(false)



  function handleEmail(e){
    setEmail(e.target.value);
  }
```



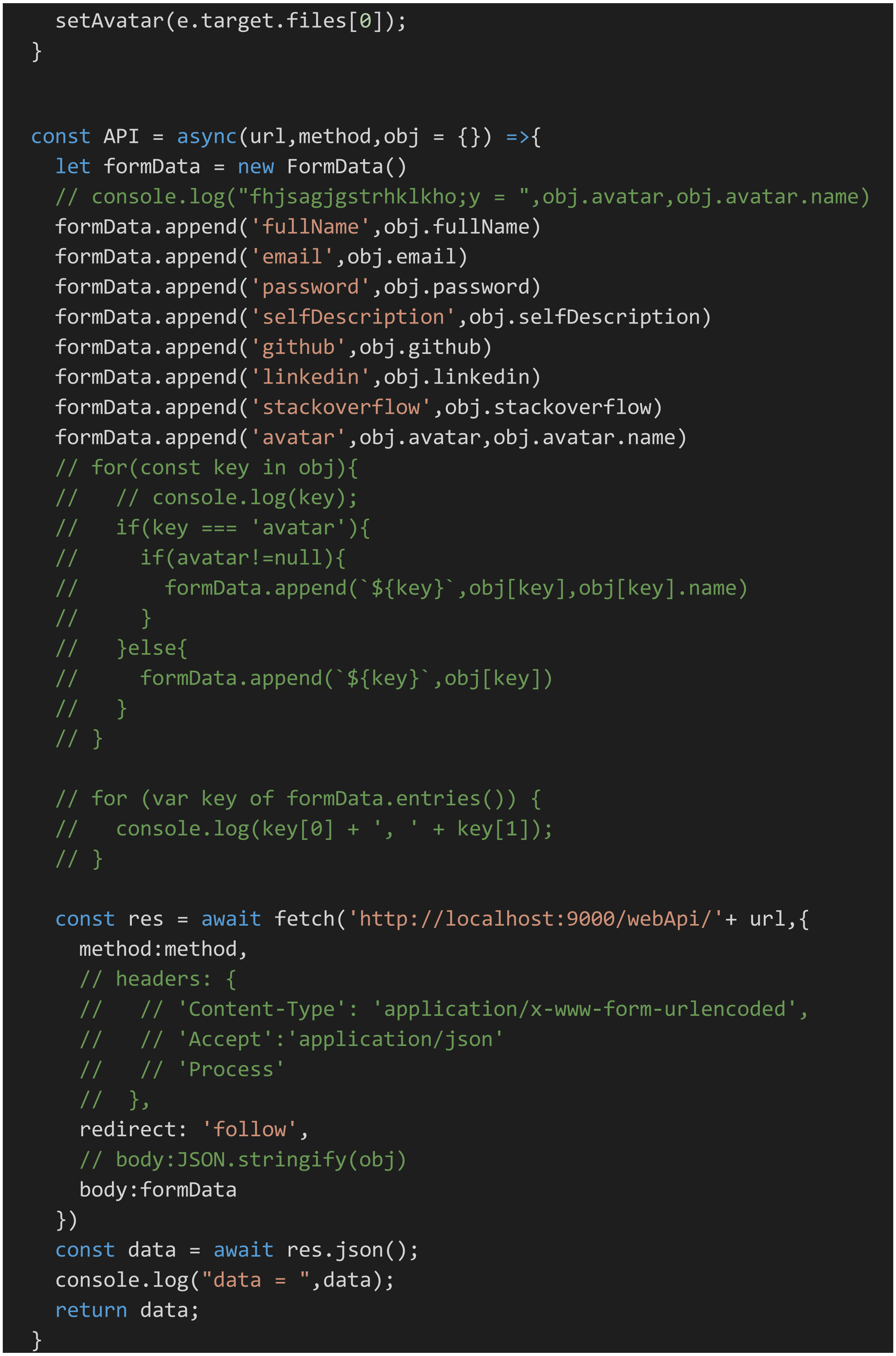
```
function handlePassword(e){
  setPassword(e.target.value);
}


const handleEvent = async() => {
  // call api, if the user valid set status = 1 and redirect to home
  // get user profile url also
  const obj = {
    email,
    password
  }
  console.log(obj);

  const userData =  await API('signin/','POST',obj);
  console.log("user data = ",userData);

  const status = userData.Data;
  if(status === 0) setStatus(0);
  else {
    setStatus(1)
    await setUserDetail(userData)
  }

  setApiCall(!apiCall)
  console.log(status);
}

useEffect(()=> {
    if(status === 0){
      alert("Username or Password is incorrect");
    }
    else if(status === 1){
      // window.location.href = '/home';
      // setUserDetail(userData)
      props.history.push({
        pathname:'/home',
          state:{userData:userData}
       });
    }
}, [apiCall])


/**
 * this will run only once when the screen renders for the first time
```

```
  useEffect(()=>{
  },[])
  */

  return (
  <div>
  <form className="form-signin" action="/thanks" method="post">
    <h1 className="h4 mb-3 font-weight-normal">Welcome Back!</h1>
    <input onChange = {handleEmail} type="email" className="form-control top" placeholder="Email"name="email"  value = {email} required autoFocus/>
    <input onChange = {handlePassword} type="password" className="form-control bottom" placeholder="Password" name="password" value = {password} required />
    <Link onClick = {handleEvent} className="btn btn-group btn-lg btn-dark btn-block" type="submit">Sign in</Link>
    <small id="signInHelpBlock" className="form-text text-muted">
    Need an account?&nbsp;&nbsp;
    <Link to="/sign-up"> Register</Link>
    </small>
  </form>
  <p className="mt-5 mb-3 text-muted text-center">&copy; Fred</p>
  <Footer />
  </div>
  )
}

export default SignIn;
```

```
import React ,{useState} from 'react';
import Footer from '../Footer.jsx';
import { Link } from 'react-router-dom';
import '../../App.css';
/**
 *
 formData.append('fullName',obj.fullName)
  formData.append('email',obj.email)
  formData.append('password',obj.password)
  formData.append('selfDescription',obj.selfDescription)
  formData.append('github',obj.github)
  formData.append('linkedin',obj.linkedin)
  formData.append('stackoverflow',obj.stackoverflow)
 */
```



```
function SignUp() {

  const [fullName, setFullName] = useState("");
  const [password, setPassword] = useState("");
  const [email, setEmail] = useState("");
  const [selfDescription, setSelfDescription] = useState("");
  const [github, setGithub] = useState("");
  const [linkedin, setLinkedin] = useState("");
  const [stackoverflow, setStackoverflow] = useState("");
  const [avatar, setAvatar] = useState(null);



  function handleFullName(e){
    setFullName(e.target.value);
  }

  function handleEmail(e){
    setEmail(e.target.value);
  }

  function handlePassword(e){
    setPassword(e.target.value);
  }

  function handleSelfDescription(e){
    setSelfDescription(e.target.value);
  }

  function handleGithub(e){
    setGithub(e.target.value);
  }

  function handleLinkedin(e){
    setLinkedin(e.target.value);
  }

  function handleStackoverflow(e){
    setStackoverflow(e.target.value);
  }

  function handleAvatar(e){
    // console.log("avtart value = ",e.target.value);
```

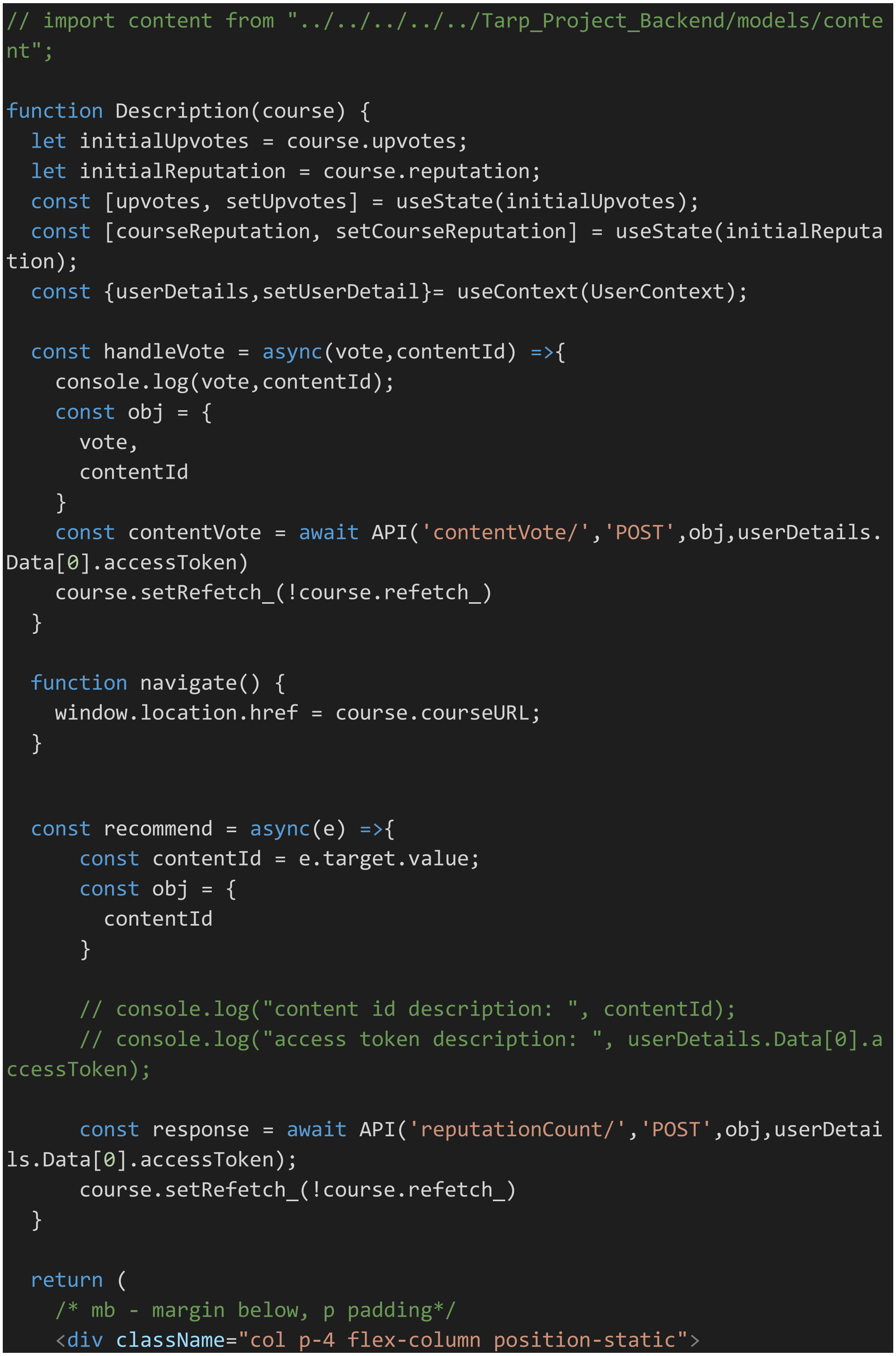


```
  setAvatar(e.target.files[0]);
}


const API = async(url,method,obj = {}) =>{
  let formData = new FormData()
  // console.log("fhjsagjgstrhklkho;y = ",obj.avatar,obj.avatar.name)
  formData.append('fullName',obj.fullName)
  formData.append('email',obj.email)
  formData.append('password',obj.password)
  formData.append('selfDescription',obj.selfDescription)
  formData.append('github',obj.github)
  formData.append('linkedin',obj.linkedin)
  formData.append('stackoverflow',obj.stackoverflow)
  formData.append('avatar',obj.avatar,obj.avatar.name)
  // for(const key in obj){
  //   // console.log(key);
  //   if(key === 'avatar'){
  //     if(avatar!=null){
  //       formData.append(`${key}`,obj[key],obj[key].name)
  //     }
  //   }else{
  //     formData.append(`${key}`,obj[key])
  //   }
  // }

  // for (var key of formData.entries()) {
  //   console.log(key[0] + ', ' + key[1]);
  // }

  const res = await fetch('http://localhost:9000/webApi/'+ url,{
    method:method,
    // headers: {
    //   // 'Content-Type': 'application/x-www-form-urlencoded',
    //   // 'Accept':'application/json'
    //   // 'Process'
    //  },
    redirect: 'follow',
    // body:JSON.stringify(obj)
    body:formData
  })
  const data = await res.json();
  console.log("data = ",data);
  return data;
}
```

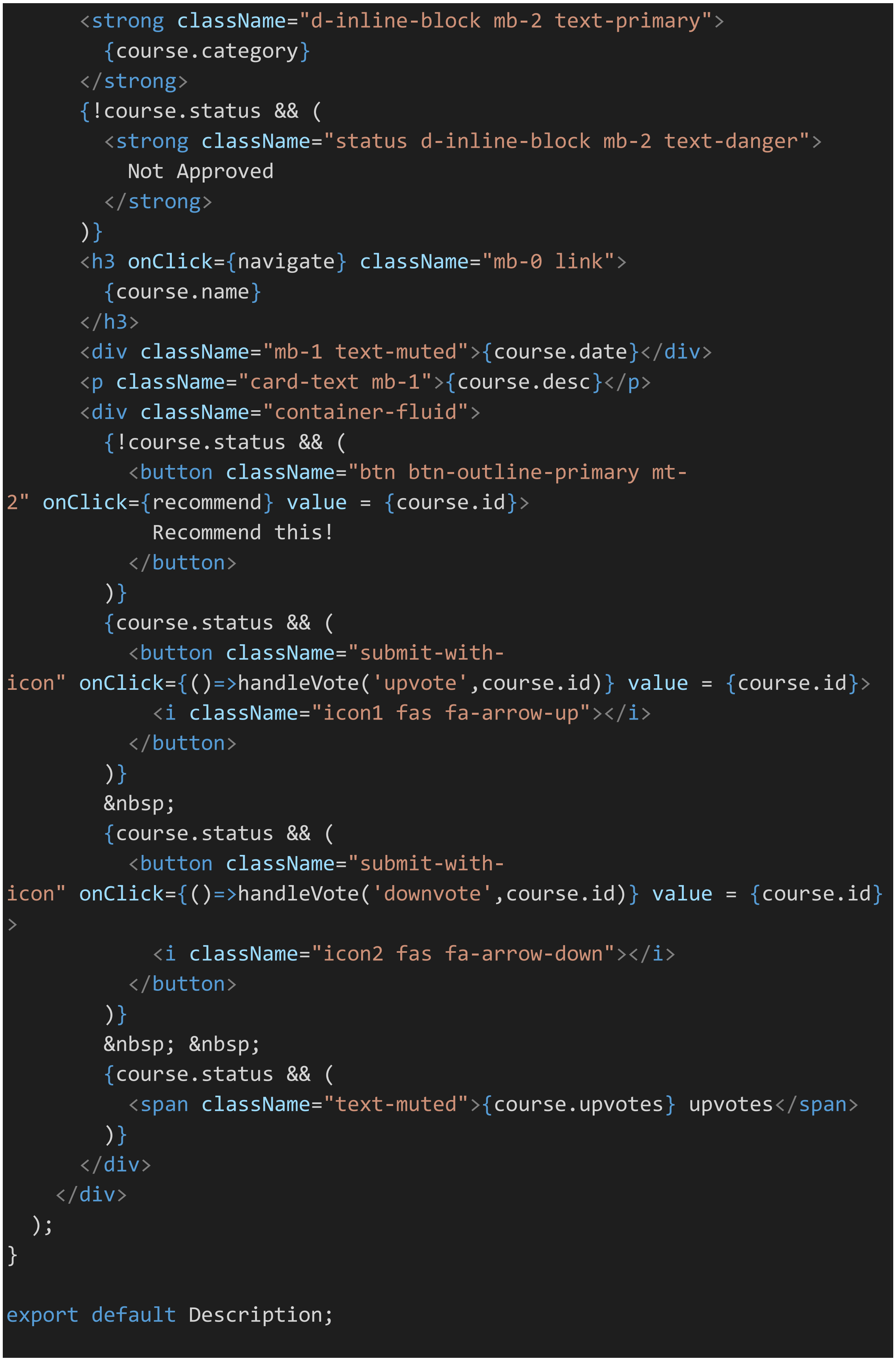
```
  const handleEvent = async() => {
    // call api, if the user valid set status = 1 and redirect to home
    // get user profile url also
    const obj = {
      fullName,
      email,
      password,
      selfDescription,
      github,
      linkedin,
      stackoverflow,
      avatar

    }
    console.log(obj);
    // const res = await fetch('http://localhost:9000/webApi/signin/',{
    //   method:'POST',
    //   headers: {
    //     'Content-Type': 'application/json'
    //    },
    //   body:JSON.stringify(obj)
    // })
    // const data = await res.json();
    // console.log("data = ",data);

    const randomdata =  await API('signup/','POST',obj);
    alert("Congrats, you have successfully signed up!")
    console.log("random data = ",randomdata);
    // setStatus(1);
  }

  return (
  <div>
  <form className="form-signin">
    <h1 className="h4 mb-3 font-weight-normal">Create an Account</h1>
    <input onChange = {handleFullName} type="text" className="form-
con-
trol top" placeholder="Username" id="fullName" name="fullName" value =
{fullName} required autoFocus/>
    <input onChange = {handleEmail} type="email" className="form-
con-
trol middle" placeholder="Email" id="email" name="email" value = {email}
 required />
```

```
    <input onChange = {handlePassword} type="password" className="form-control middle" placeholder="Password" id = "password" name="password" value = {password} required />
    <input onChange = {handleSelfDescription} value = {selfDescription} type="text" className="form-control middle" placeholder="Self Description" id="selfDescription" name="selfDescription"/>
    <input onChange = {handleGithub} value = {github} type="text" className="form-control middle" placeholder="Github Account" id="github" name="github"/>
    <input onChange = {handleLinkedin} value = {linkedin} type="text" className="form-control middle" placeholder="Linkedin Account" id="linkedin" name="linkedin"/>
    <input onChange = {handleStackoverflow} value = {stackoverflow} type="text" className="form-control bottom" placeholder="Stackoverflow Account" id="stackoverflow" name="stackoverflow"/>
    <input onChange = {handleAvatar} type="file" />
    <Link onClick = {handleEvent} className="btn btn-group btn-lg btn-dark btn-block" type="submit">Continue</Link>
    <small id="signInHelpBlock" className="form-text text-muted">
    <Link to="/sign-in">Already have an account?</Link>
    </small>
  </form>
  <p className="mt-5 mb-3 text-muted text-center">&copy; Fred</p>
  <Footer />
  </div>
  )
}

export default SignUp;
```

```
import React, { useState, useContext } from "react";
import "../styles.css";
import {API} from '../../../ApiSchema';
import {UserContext} from '../../../userContext';
```

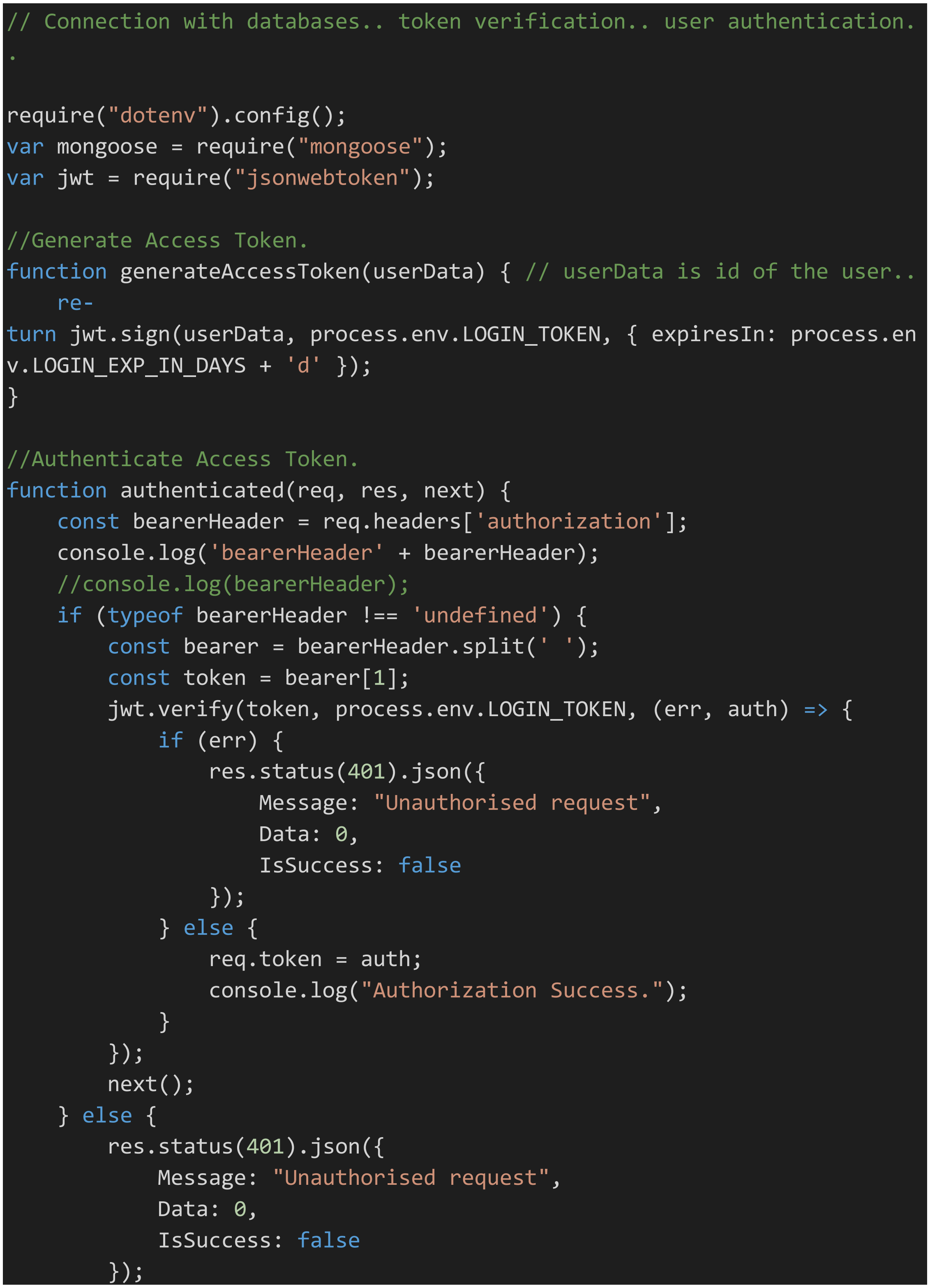


```
// import content from "../../../../../Tarp_Project_Backend/models/content";

function Description(course) {
  let initialUpvotes = course.upvotes;
  let initialReputation = course.reputation;
  const [upvotes, setUpvotes] = useState(initialUpvotes);
  const [courseReputation, setCourseReputation] = useState(initialReputation);
  const {userDetails,setUserDetail}= useContext(UserContext);

  const handleVote = async(vote,contentId) =>{
    console.log(vote,contentId);
    const obj = {
      vote,
      contentId
    }
    const contentVote = await API('contentVote/','POST',obj,userDetails.Data[0].accessToken)
    course.setRefetch_(!course.refetch_)
  }

  function navigate() {
    window.location.href = course.courseURL;
  }


  const recommend = async(e) =>{
      const contentId = e.target.value;
      const obj = {
        contentId
      }

      // console.log("content id description: ", contentId);
      // console.log("access token description: ", userDetails.Data[0].accessToken);

      const response = await API('reputationCount/','POST',obj,userDetails.Data[0].accessToken);
      course.setRefetch_(!course.refetch_)
  }

  return (
    /* mb - margin below, p padding*/
    <div className="col p-4 flex-column position-static">
```



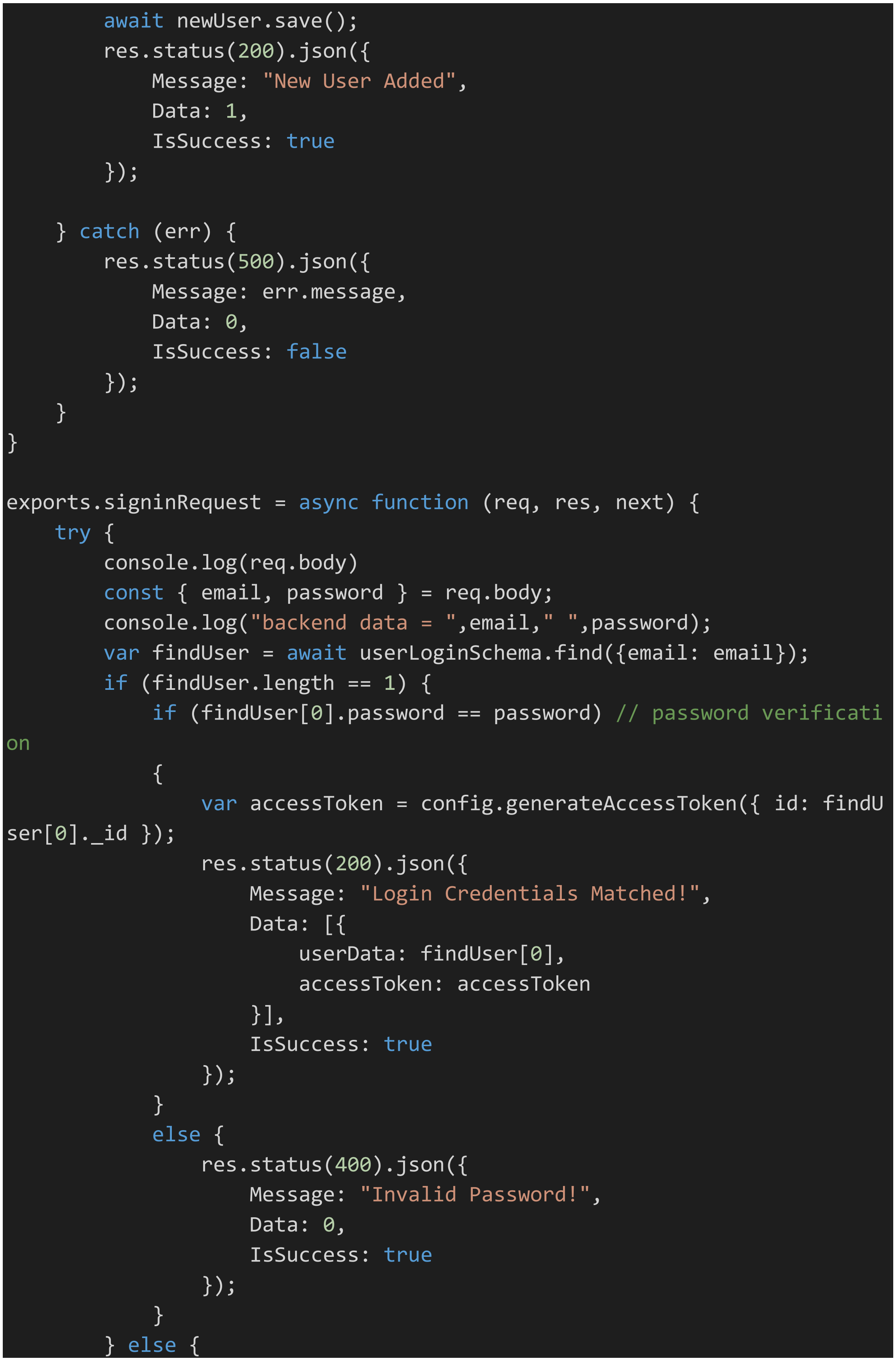
```
      <strong className="d-inline-block mb-2 text-primary">
        {course.category}
      </strong>
      {!course.status && (
        <strong className="status d-inline-block mb-2 text-danger">
          Not Approved
        </strong>
      )}
      <h3 onClick={navigate} className="mb-0 link">
        {course.name}
      </h3>
      <div className="mb-1 text-muted">{course.date}</div>
      <p className="card-text mb-1">{course.desc}</p>
      <div className="container-fluid">
        {!course.status && (
          <button className="btn btn-outline-primary mt-
2" onClick={recommend} value = {course.id}>
            Recommend this!
          </button>
        )}
        {course.status && (
          <button className="submit-with-
icon" onClick={()=>handleVote('upvote',course.id)} value = {course.id}>
            <i className="icon1 fas fa-arrow-up"></i>
          </button>
        )}
        &nbsp;
        {course.status && (
          <button className="submit-with-
icon" onClick={()=>handleVote('downvote',course.id)} value = {course.id}
>
            <i className="icon2 fas fa-arrow-down"></i>
          </button>
        )}
        &nbsp; &nbsp;
        {course.status && (
          <span className="text-muted">{course.upvotes} upvotes</span>
        )}
      </div>
    </div>
  );
}

export default Description;
```

SAMPLE BACKEND CODE:

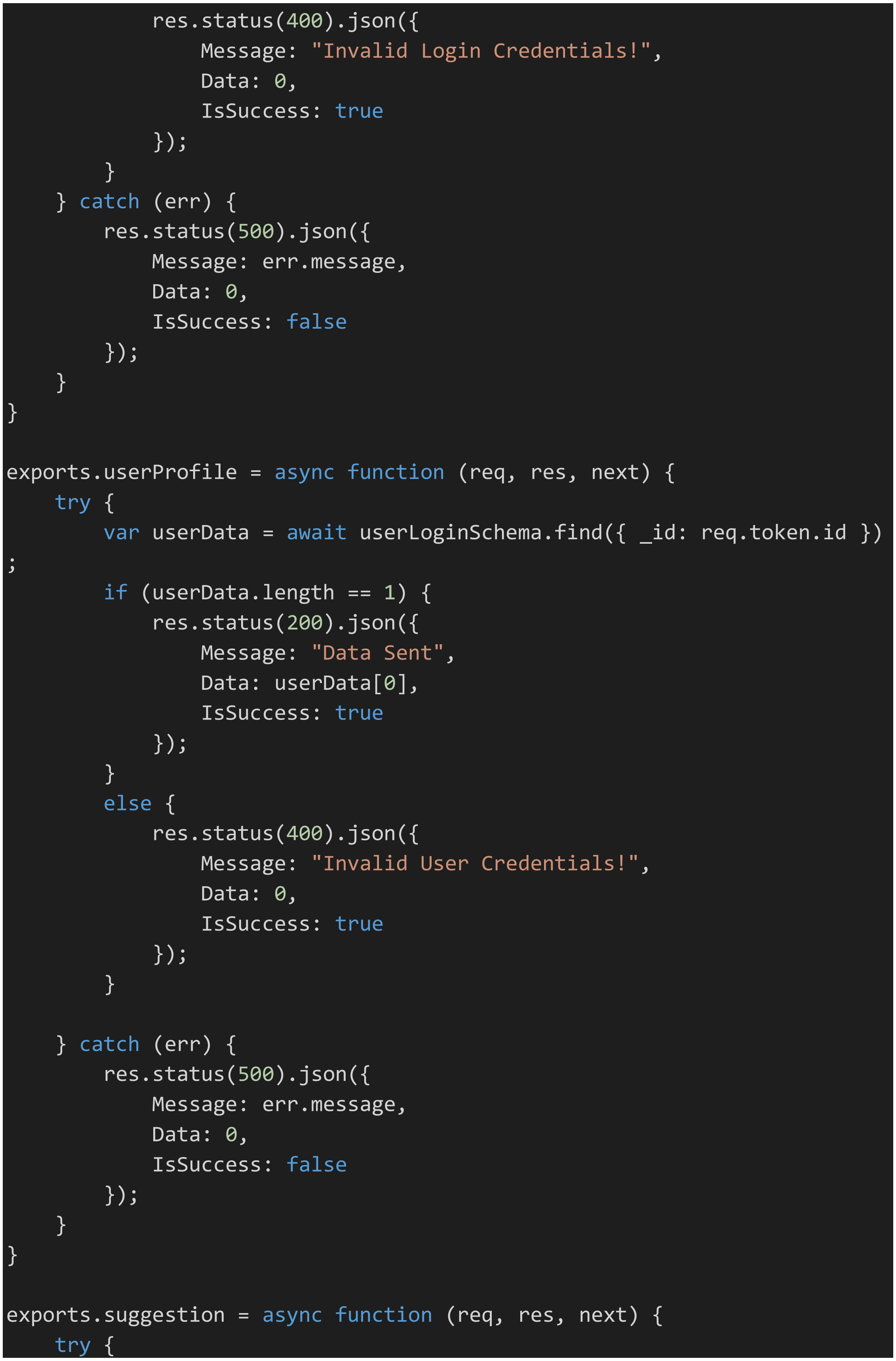


```
// Connection with databases.. token verification.. user authentication...

require("dotenv").config();
var mongoose = require("mongoose");
var jwt = require("jsonwebtoken");

//Generate Access Token.
function generateAccessToken(userData) { // userData is id of the user..
    return jwt.sign(userData, process.env.LOGIN_TOKEN, { expiresIn: process.env.LOGIN_EXP_IN_DAYS + 'd' });
}

//Authenticate Access Token.
function authenticated(req, res, next) {
    const bearerHeader = req.headers['authorization'];
    console.log('bearerHeader' + bearerHeader);
    //console.log(bearerHeader);
    if (typeof bearerHeader !== 'undefined') {
        const bearer = bearerHeader.split(' ');
        const token = bearer[1];
        jwt.verify(token, process.env.LOGIN_TOKEN, (err, auth) => {
            if (err) {
                res.status(401).json({
                    Message: "Unauthorised request",
                    Data: 0,
                    IsSuccess: false
                });
            } else {
                req.token = auth;
                console.log("Authorization Success.");
            }
        });
        next();
    } else {
        res.status(401).json({
            Message: "Unauthorised request",
            Data: 0,
            IsSuccess: false
        });
    }
}

//Connecting with mongoDB Database
mongoose.connect(process.env.MONGO_URL, { useNewUrlParser: true, useUnifiedTopology: true, useFindAndModify: false });
mongoose.connection
    .once('open', () => console.log("Well done! , connected with mongoDB database"))
    .on('error', error => console.log("Opps! database connection error:" + error));

module.exports = {
    generateAccessToken,
    authenticated,
    mongoose
}
```

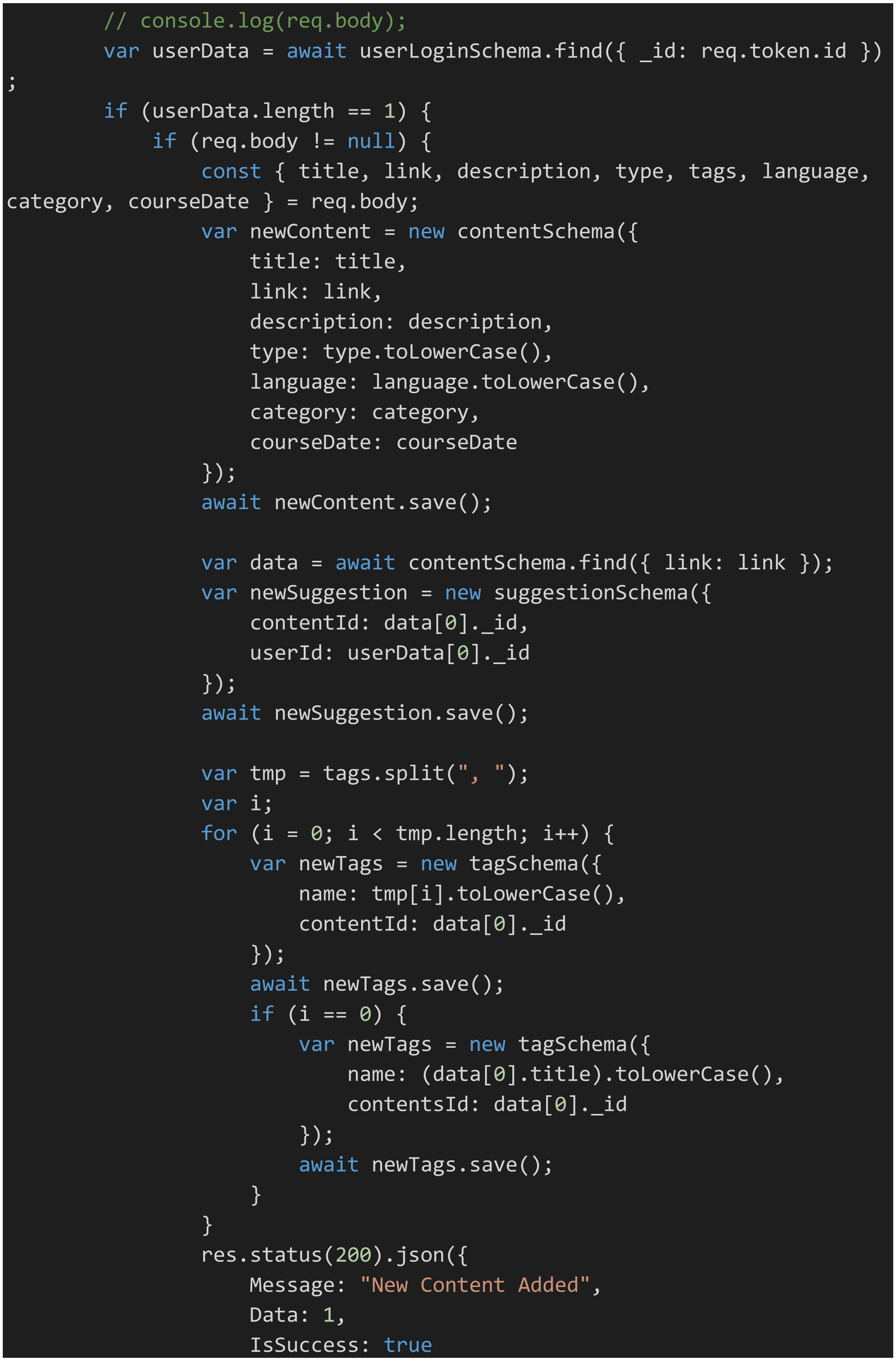
```
var config = require("../config");

var userLoginSchema = require("../models/userDetails");
var contentSchema = require("../models/content");
var tagSchema = require("../models/tags");
var voteSchema = require("../models/votes");
var suggestionSchema = require("../models/suggestion");
var upvoteSchema = require("../models/upvote");
var countSchema = require("../models/counts");

exports.signupRequest = async function (req, res, next) {
    console.log("overe here");
    console.log("body = ",req.body)
    // console.log("file = ",req.files['avatar'])
    try {
        var newUser = new userLoginSchema({
            fullName: req.body.fullName,
            email: req.body.email,
            password: req.body.password,
            selfDescription: req.body.selfDescription,
            github: req.body.github,
            linkedin: req.body.linkedin,
            stackoverflow: req.body.stackoverflow,
            profileImg: req.files != null ? req.files['avatar'][0].path : null
        });
```



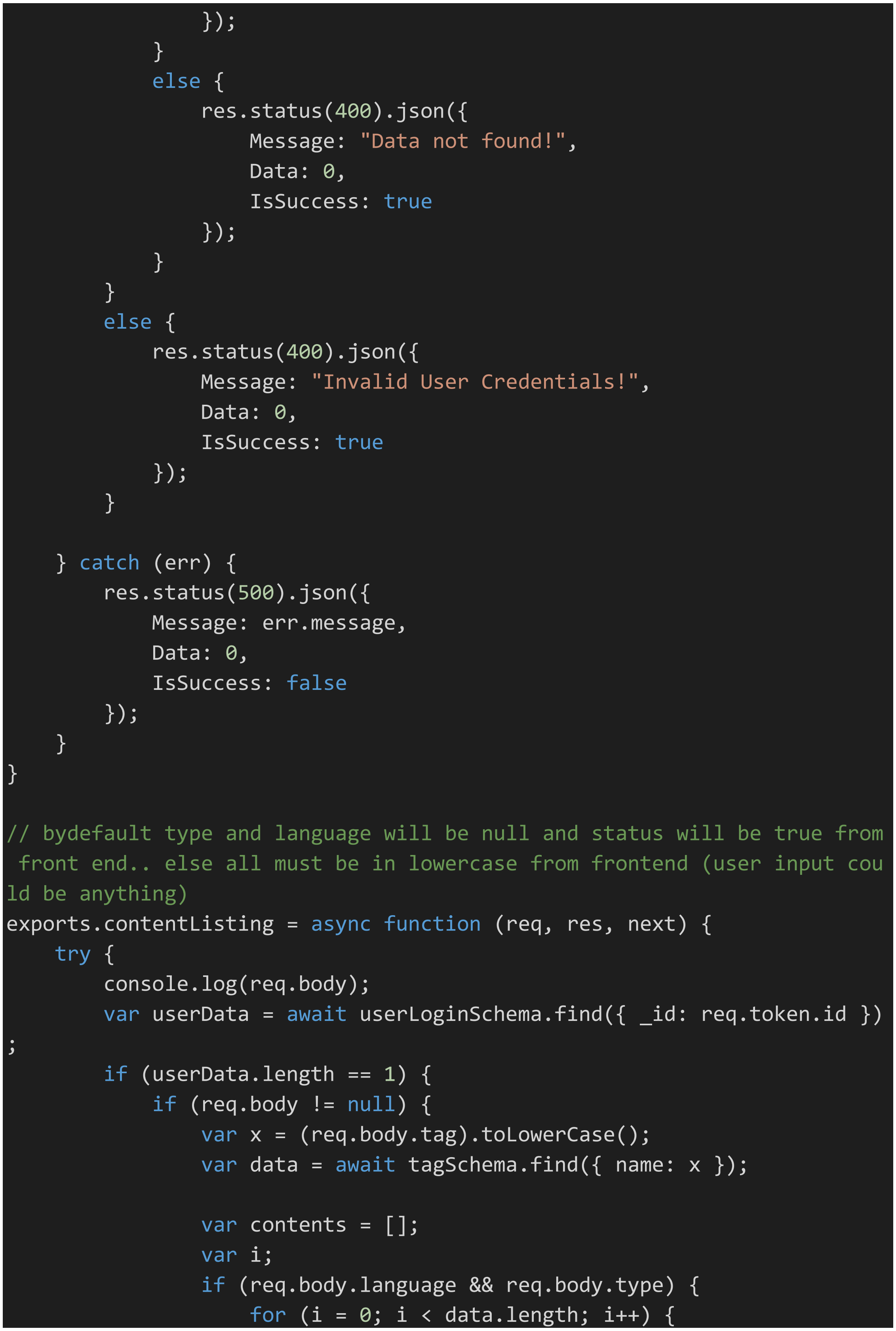
```
        await newUser.save();
        res.status(200).json({
            Message: "New User Added",
            Data: 1,
            IsSuccess: true
        });

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.signinRequest = async function (req, res, next) {
    try {
        console.log(req.body)
        const { email, password } = req.body;
        console.log("backend data = ",email," ",password);
        var findUser = await userLoginSchema.find({email: email});
        if (findUser.length == 1) {
            if (findUser[0].password == password) // password verificati
on
            {
                var accessToken = config.generateAccessToken({ id: findU
ser[0]._id });
                res.status(200).json({
                    Message: "Login Credentials Matched!",
                    Data: [{
                        userData: findUser[0],
                        accessToken: accessToken
                    }],
                    IsSuccess: true
                });
            }
            else {
                res.status(400).json({
                    Message: "Invalid Password!",
                    Data: 0,
                    IsSuccess: true
                });
            }
        } else {
```



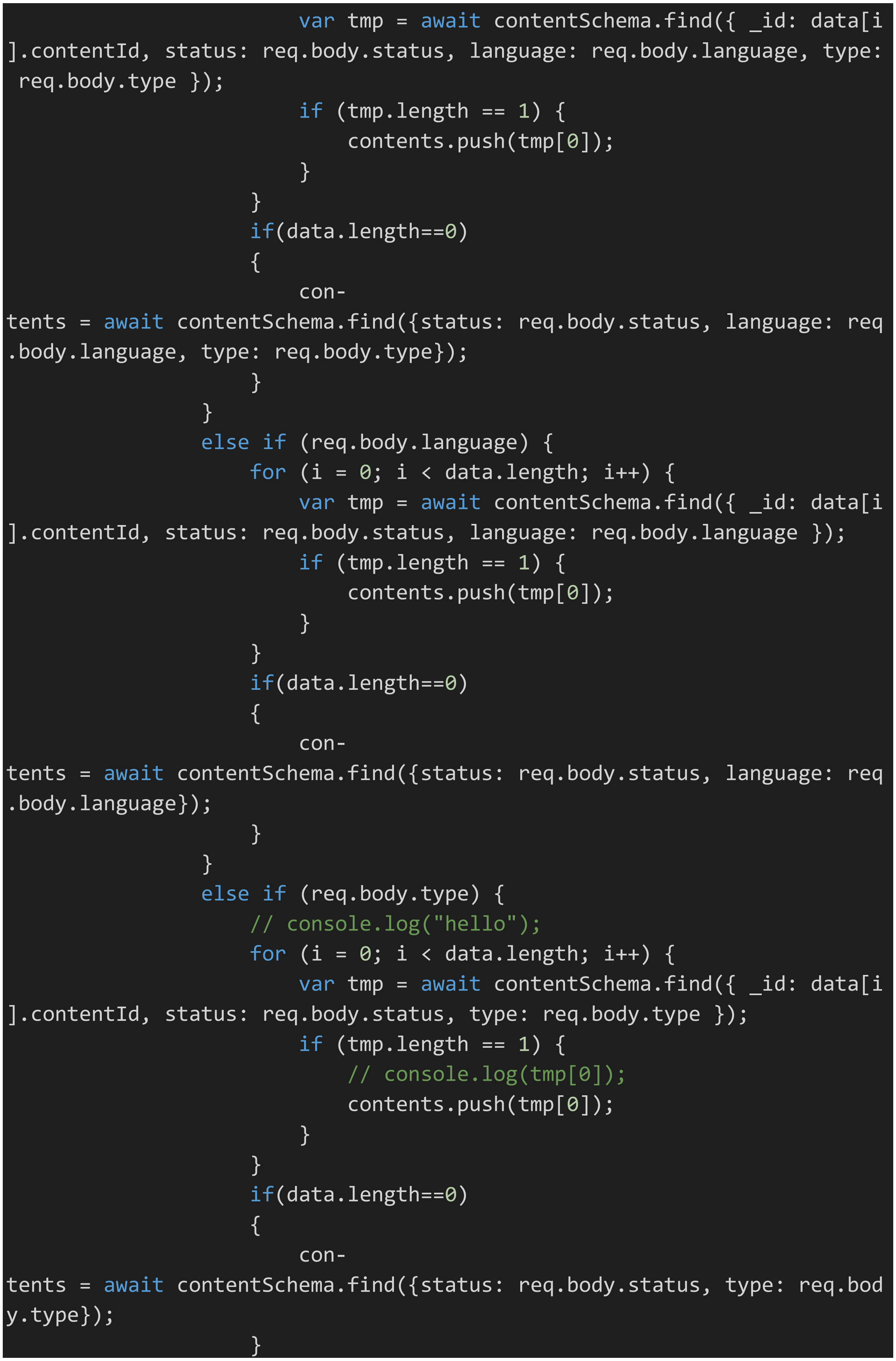
```
            res.status(400).json({
                Message: "Invalid Login Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }
    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.userProfile = async function (req, res, next) {
    try {
        var userData = await userLoginSchema.find({ _id: req.token.id })
;
        if (userData.length == 1) {
            res.status(200).json({
                Message: "Data Sent",
                Data: userData[0],
                IsSuccess: true
            });
        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.suggestion = async function (req, res, next) {
    try {
```



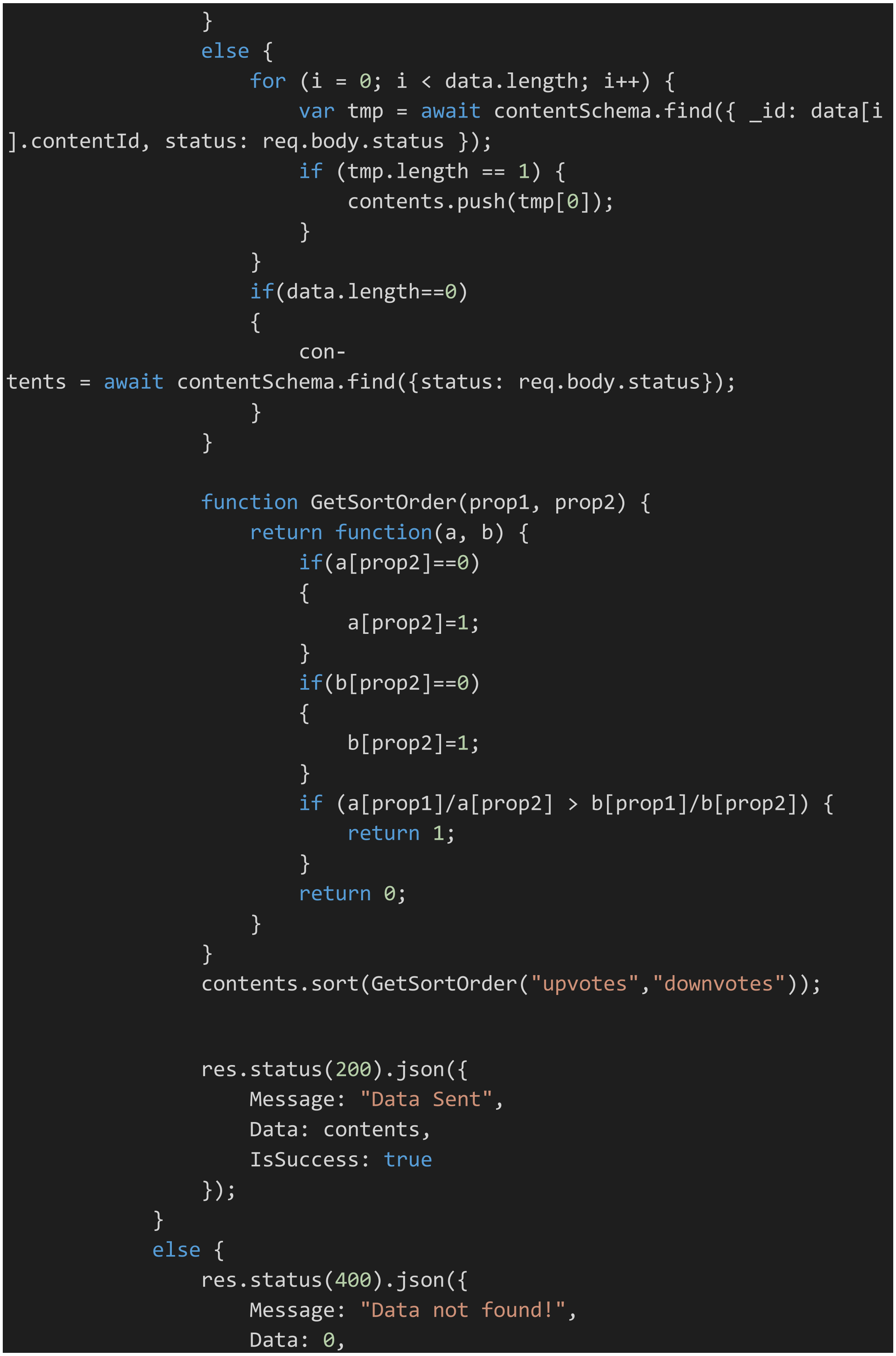
```
        // console.log(req.body);
        var userData = await userLoginSchema.find({ _id: req.token.id })
;
        if (userData.length == 1) {
            if (req.body != null) {
                const { title, link, description, type, tags, language,
category, courseDate } = req.body;
                var newContent = new contentSchema({
                    title: title,
                    link: link,
                    description: description,
                    type: type.toLowerCase(),
                    language: language.toLowerCase(),
                    category: category,
                    courseDate: courseDate
                });
                await newContent.save();

                var data = await contentSchema.find({ link: link });
                var newSuggestion = new suggestionSchema({
                    contentId: data[0]._id,
                    userId: userData[0]._id
                });
                await newSuggestion.save();

                var tmp = tags.split(", ");
                var i;
                for (i = 0; i < tmp.length; i++) {
                    var newTags = new tagSchema({
                        name: tmp[i].toLowerCase(),
                        contentId: data[0]._id
                    });
                    await newTags.save();
                    if (i == 0) {
                        var newTags = new tagSchema({
                            name: (data[0].title).toLowerCase(),
                            contentsId: data[0]._id
                        });
                        await newTags.save();
                    }
                }
                res.status(200).json({
                    Message: "New Content Added",
                    Data: 1,
                    IsSuccess: true
```



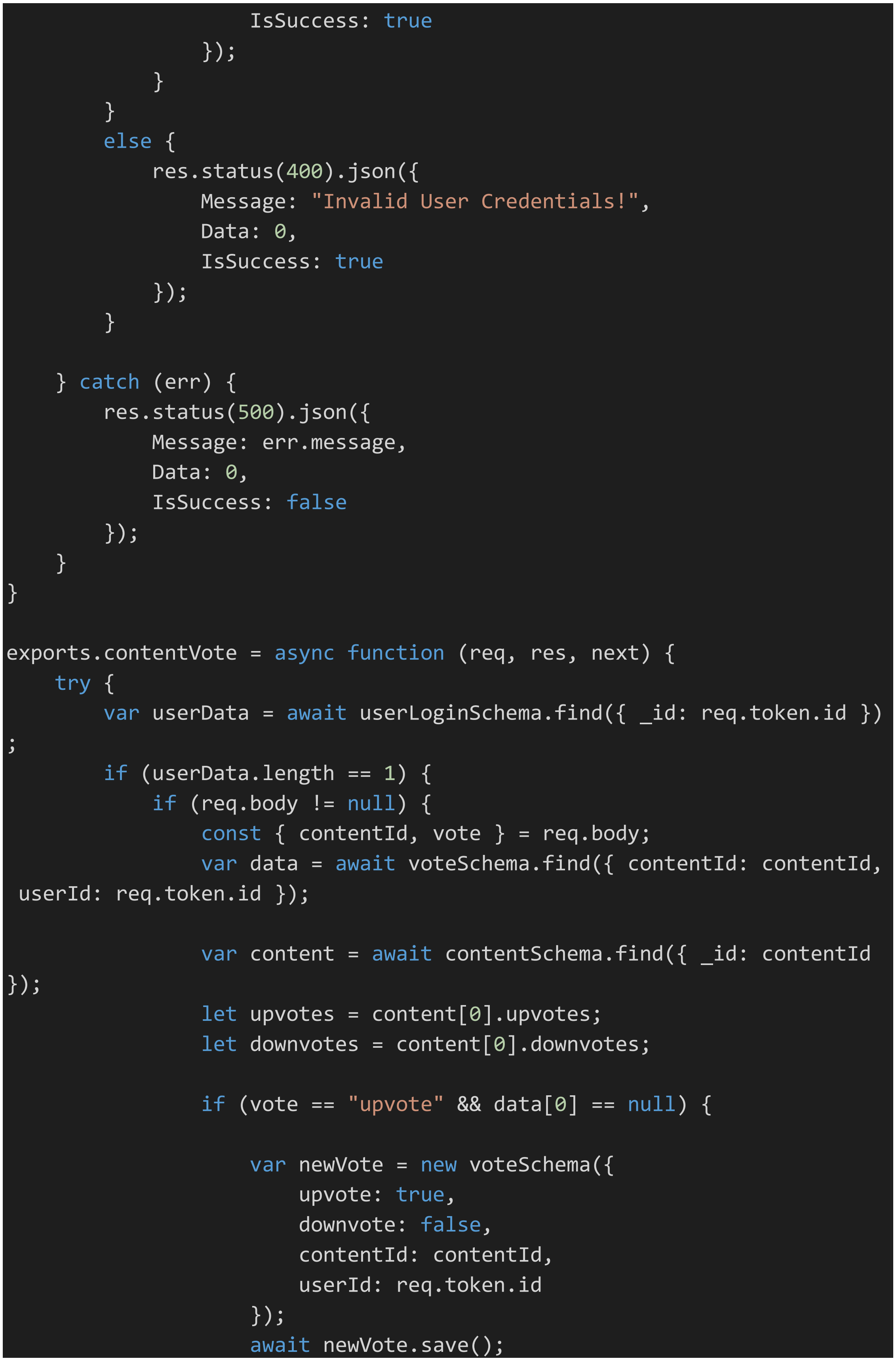
```
                });
            }
            else {
                res.status(400).json({
                    Message: "Data not found!",
                    Data: 0,
                    IsSuccess: true
                });
            }
        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

// bydefault type and language will be null and status will be true from
 front end.. else all must be in lowercase from frontend (user input cou
ld be anything)
exports.contentListing = async function (req, res, next) {
    try {
        console.log(req.body);
        var userData = await userLoginSchema.find({ _id: req.token.id })
;
        if (userData.length == 1) {
            if (req.body != null) {
                var x = (req.body.tag).toLowerCase();
                var data = await tagSchema.find({ name: x });

                var contents = [];
                var i;
                if (req.body.language && req.body.type) {
                    for (i = 0; i < data.length; i++) {
```



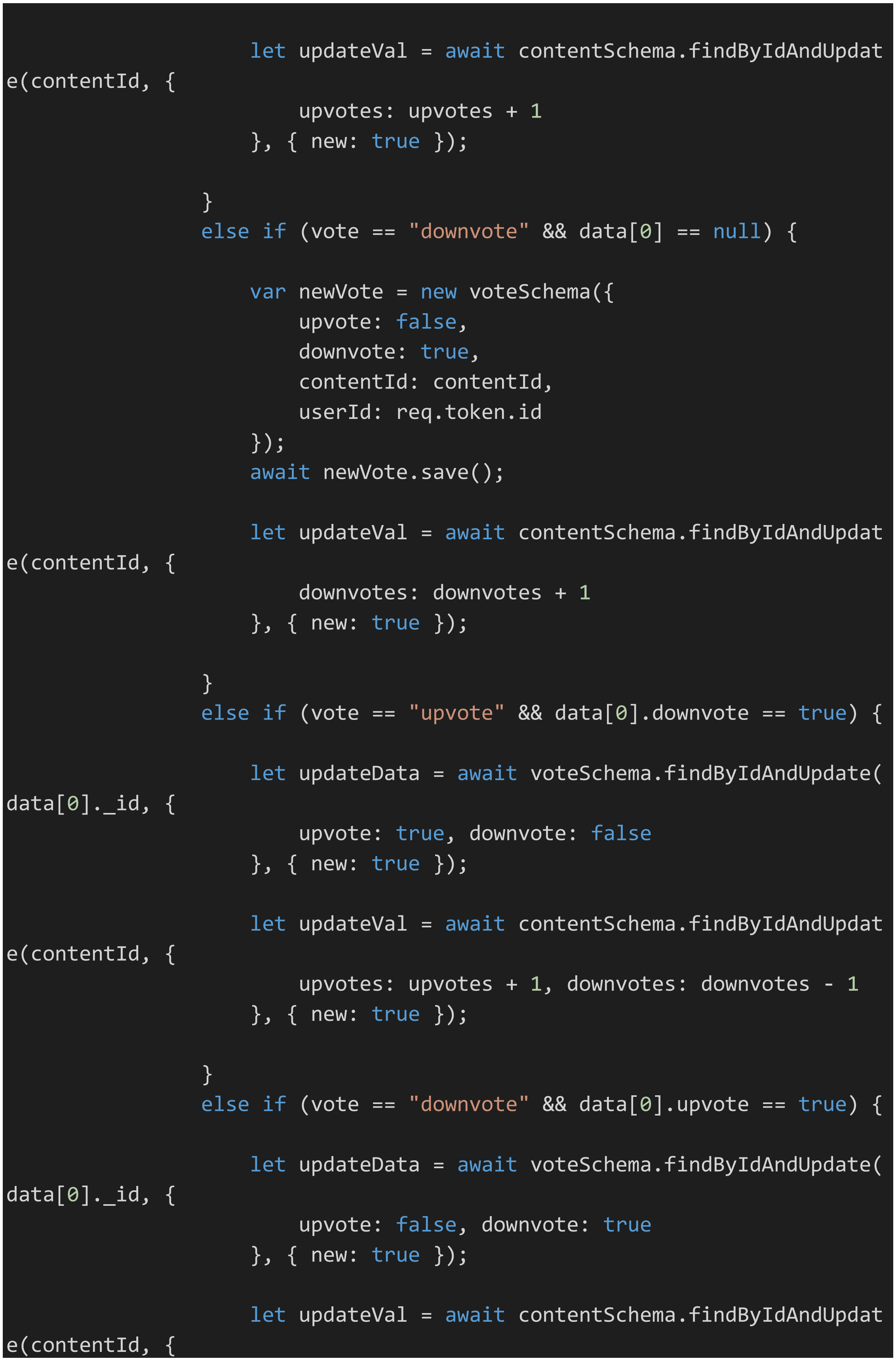
```
                        var tmp = await contentSchema.find({ _id: data[i].contentId, status: req.body.status, language: req.body.language, type: req.body.type });
                        if (tmp.length == 1) {
                            contents.push(tmp[0]);
                        }
                    }
                    if(data.length==0)
                    {
                        contents = await contentSchema.find({status: req.body.status, language: req.body.language, type: req.body.type});
                    }
                }
                else if (req.body.language) {
                    for (i = 0; i < data.length; i++) {
                        var tmp = await contentSchema.find({ _id: data[i].contentId, status: req.body.status, language: req.body.language });
                        if (tmp.length == 1) {
                            contents.push(tmp[0]);
                        }
                    }
                    if(data.length==0)
                    {
                        contents = await contentSchema.find({status: req.body.status, language: req.body.language});
                    }
                }
                else if (req.body.type) {
                    // console.log("hello");
                    for (i = 0; i < data.length; i++) {
                        var tmp = await contentSchema.find({ _id: data[i].contentId, status: req.body.status, type: req.body.type });
                        if (tmp.length == 1) {
                            // console.log(tmp[0]);
                            contents.push(tmp[0]);
                        }
                    }
                    if(data.length==0)
                    {
                        contents = await contentSchema.find({status: req.body.status, type: req.body.type});
                    }
```



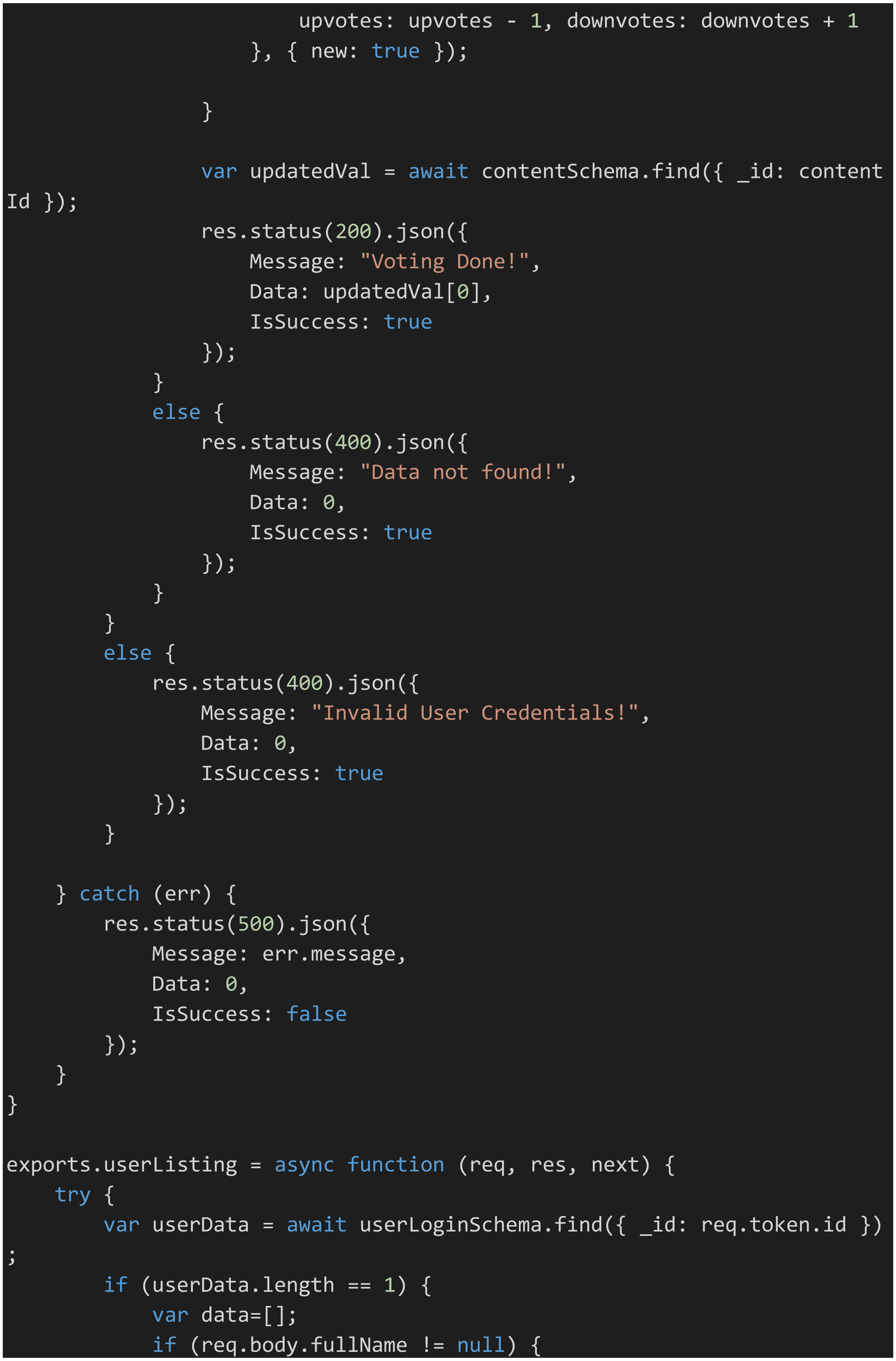
```
                }
                else {
                    for (i = 0; i < data.length; i++) {
                        var tmp = await contentSchema.find({ _id: data[i
].contentId, status: req.body.status });
                        if (tmp.length == 1) {
                            contents.push(tmp[0]);
                        }
                    }
                    if(data.length==0)
                    {
                        contents = await contentSchema.find({status: req.body.status});
                    }
                }

                function GetSortOrder(prop1, prop2) {
                    return function(a, b) {
                        if(a[prop2]==0)
                        {
                            a[prop2]=1;
                        }
                        if(b[prop2]==0)
                        {
                            b[prop2]=1;
                        }
                        if (a[prop1]/a[prop2] > b[prop1]/b[prop2]) {
                            return 1;
                        }
                        return 0;
                    }
                }
                contents.sort(GetSortOrder("upvotes","downvotes"));


                res.status(200).json({
                    Message: "Data Sent",
                    Data: contents,
                    IsSuccess: true
                });
            }
            else {
                res.status(400).json({
                    Message: "Data not found!",
                    Data: 0,
```



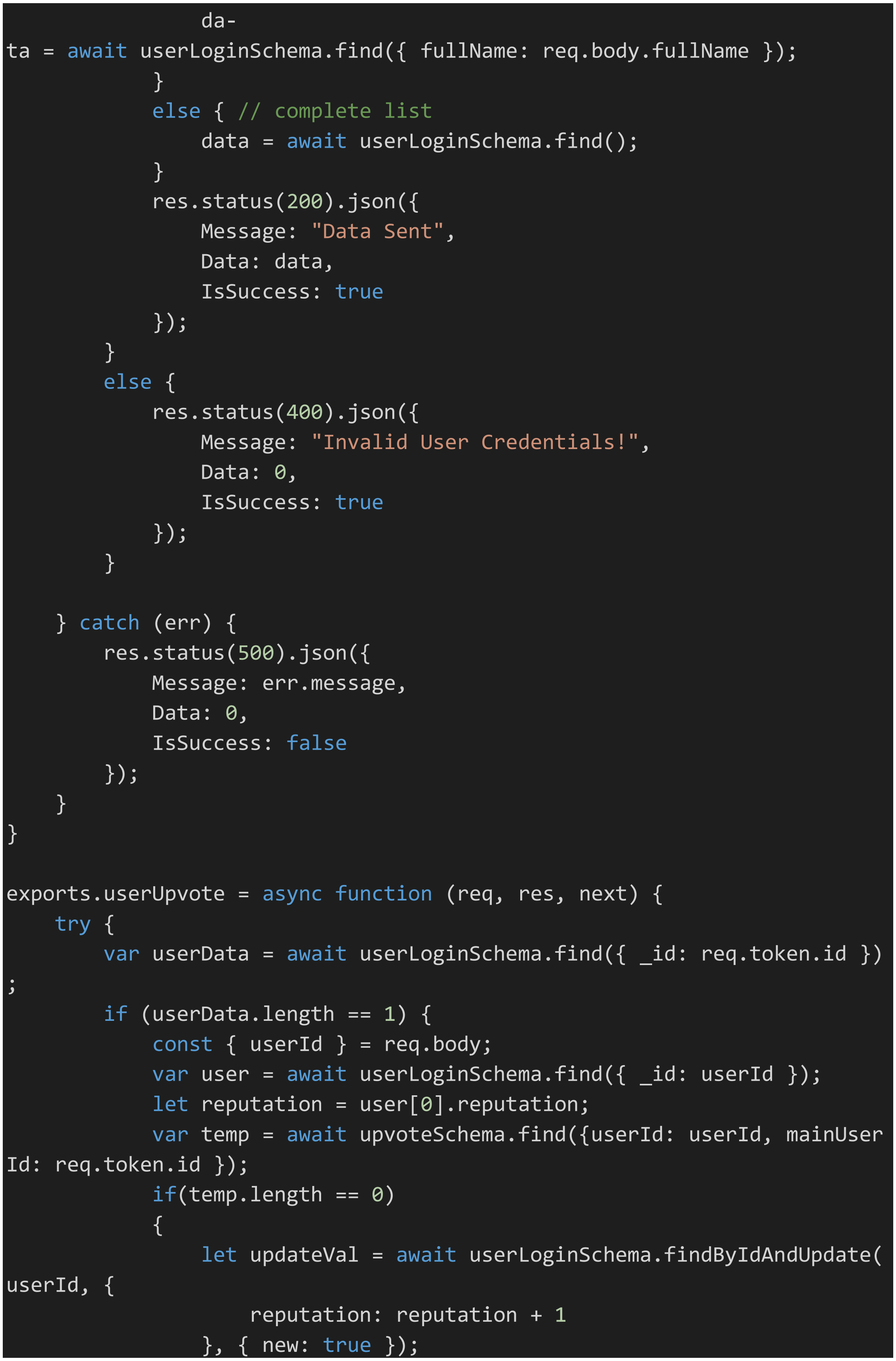
```
                    IsSuccess: true
                });
            }
        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.contentVote = async function (req, res, next) {
    try {
        var userData = await userLoginSchema.find({ _id: req.token.id })
;
        if (userData.length == 1) {
            if (req.body != null) {
                const { contentId, vote } = req.body;
                var data = await voteSchema.find({ contentId: contentId,
 userId: req.token.id });

                var content = await contentSchema.find({ _id: contentId
});
                let upvotes = content[0].upvotes;
                let downvotes = content[0].downvotes;

                if (vote == "upvote" && data[0] == null) {

                    var newVote = new voteSchema({
                        upvote: true,
                        downvote: false,
                        contentId: contentId,
                        userId: req.token.id
                    });
                    await newVote.save();
```



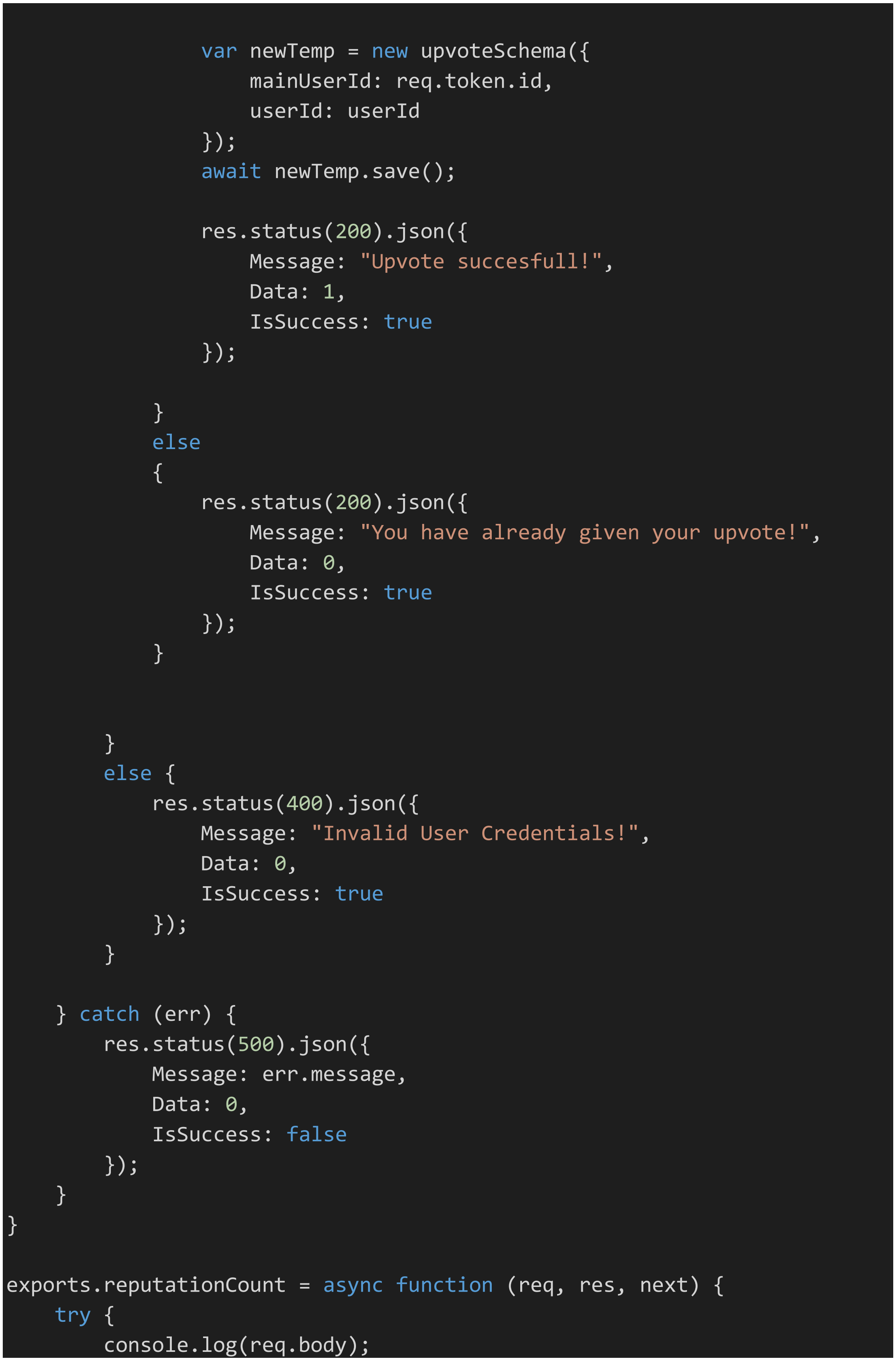
```
                    let updateVal = await contentSchema.findByIdAndUpdat
e(contentId, {
                        upvotes: upvotes + 1
                    }, { new: true });

                }
                else if (vote == "downvote" && data[0] == null) {

                    var newVote = new voteSchema({
                        upvote: false,
                        downvote: true,
                        contentId: contentId,
                        userId: req.token.id
                    });
                    await newVote.save();

                    let updateVal = await contentSchema.findByIdAndUpdat
e(contentId, {
                        downvotes: downvotes + 1
                    }, { new: true });

                }
                else if (vote == "upvote" && data[0].downvote == true) {

                    let updateData = await voteSchema.findByIdAndUpdate(
data[0]._id, {
                        upvote: true, downvote: false
                    }, { new: true });

                    let updateVal = await contentSchema.findByIdAndUpdat
e(contentId, {
                        upvotes: upvotes + 1, downvotes: downvotes - 1
                    }, { new: true });

                }
                else if (vote == "downvote" && data[0].upvote == true) {

                    let updateData = await voteSchema.findByIdAndUpdate(
data[0]._id, {
                        upvote: false, downvote: true
                    }, { new: true });

                    let updateVal = await contentSchema.findByIdAndUpdat
e(contentId, {
```



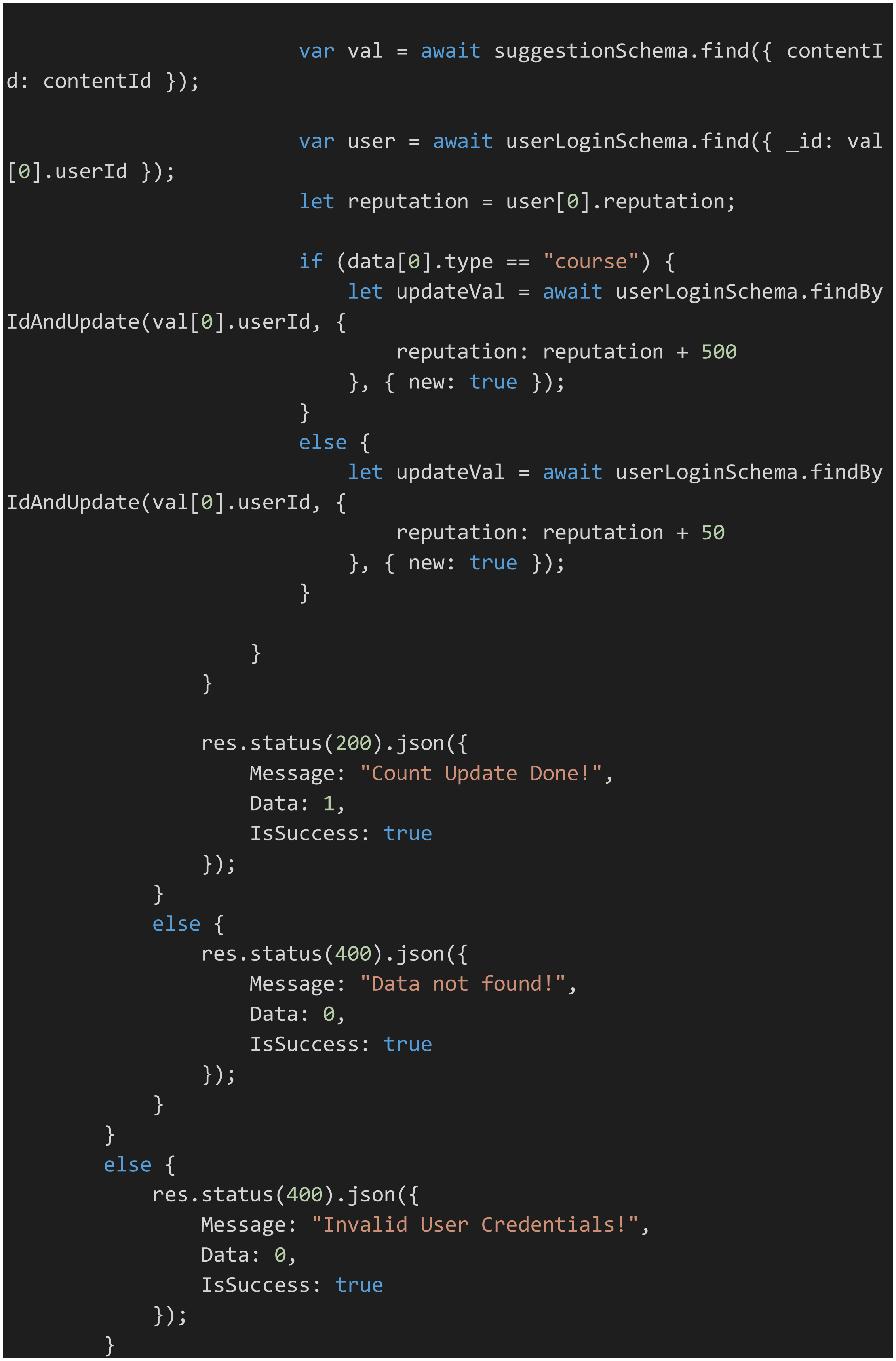
```
                        upvotes: upvotes - 1, downvotes: downvotes + 1
                    }, { new: true });

                }

                var updatedVal = await contentSchema.find({ _id: content
Id });
                res.status(200).json({
                    Message: "Voting Done!",
                    Data: updatedVal[0],
                    IsSuccess: true
                });
            }
            else {
                res.status(400).json({
                    Message: "Data not found!",
                    Data: 0,
                    IsSuccess: true
                });
            }
        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.userListing = async function (req, res, next) {
    try {
        var userData = await userLoginSchema.find({ _id: req.token.id })
;
        if (userData.length == 1) {
            var data=[];
            if (req.body.fullName != null) {
```



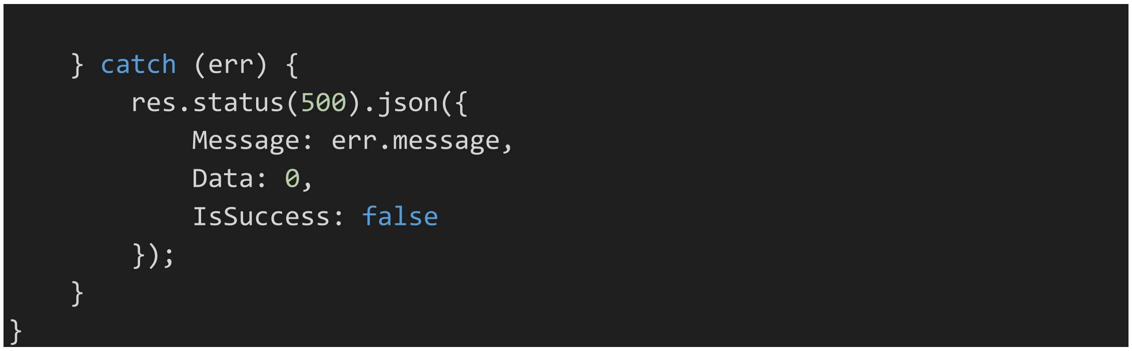
```
                da-
ta = await userLoginSchema.find({ fullName: req.body.fullName });
            }
            else { // complete list
                data = await userLoginSchema.find();
            }
            res.status(200).json({
                Message: "Data Sent",
                Data: data,
                IsSuccess: true
            });
        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.userUpvote = async function (req, res, next) {
    try {
        var userData = await userLoginSchema.find({ _id: req.token.id })
;
        if (userData.length == 1) {
            const { userId } = req.body;
            var user = await userLoginSchema.find({ _id: userId });
            let reputation = user[0].reputation;
            var temp = await upvoteSchema.find({userId: userId, mainUser
Id: req.token.id });
            if(temp.length == 0)
            {
                let updateVal = await userLoginSchema.findByIdAndUpdate(
userId, {
                    reputation: reputation + 1
                }, { new: true });
```



```
                var newTemp = new upvoteSchema({
                    mainUserId: req.token.id,
                    userId: userId
                });
                await newTemp.save();

                res.status(200).json({
                    Message: "Upvote succesfull!",
                    Data: 1,
                    IsSuccess: true
                });

            }
            else
            {
                res.status(200).json({
                    Message: "You have already given your upvote!",
                    Data: 0,
                    IsSuccess: true
                });
            }


        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }

    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}

exports.reputationCount = async function (req, res, next) {
    try {
        console.log(req.body);
        console.log(req.token);
        var userData = await userLoginSchema.find({ _id: req.token.id });
        if (userData.length == 1) {
            if (req.body != null) {
                const { contentId } = req.body;

                var content = await contentSchema.find({ _id: contentId });
                // console.log(content[0]);
                let reputationCount = content[0].reputationCount;

                if (userData[0].fullName == "Admin123") {
                    let updateVal = await contentSchema.findByIdAndUpdate(contentId, {
                        reputationCount: reputationCount + 5000, status: true
                    }, { new: true });
                }
                else {
                    var temp = await countSchema.find({contentId: contentId, userId: req.token.id });
                    if(temp.length == 0)
                    {
                        let updateVal = await contentSchema.findByIdAndUpdate(contentId, {
                            reputationCount: reputationCount + Math.min(userData[0].reputation,500)
                        }, { new: true });

                        var newTemp = new countSchema({
                            contentId: contentId,
                            userId: req.token.id
                        });
                        await newTemp.save();
                    }


                    var data = await contentSchema.find({ _id: contentId });
                    if (data[0].reputationCount >= 5000) {
                        let updateVal = await contentSchema.findByIdAndUpdate(contentId, {
                            status: true
                        }, { new: true });
```



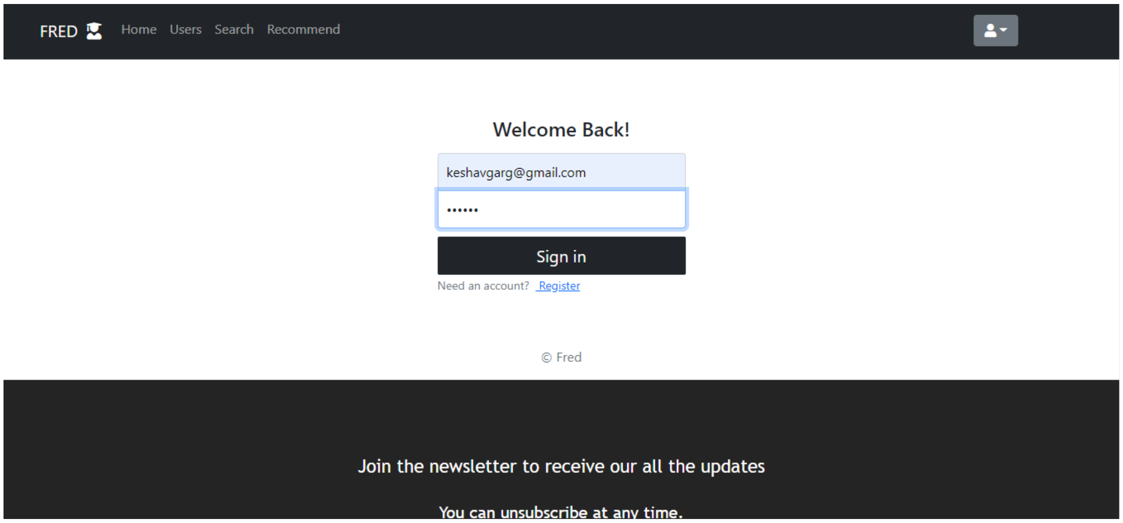
```
                        var val = await suggestionSchema.find({ contentI
d: contentId });

                        var user = await userLoginSchema.find({ _id: val
[0].userId });
                        let reputation = user[0].reputation;

                        if (data[0].type == "course") {
                            let updateVal = await userLoginSchema.findBy
IdAndUpdate(val[0].userId, {
                                reputation: reputation + 500
                            }, { new: true });
                        }
                        else {
                            let updateVal = await userLoginSchema.findBy
IdAndUpdate(val[0].userId, {
                                reputation: reputation + 50
                            }, { new: true });
                        }

                    }
                }

                res.status(200).json({
                    Message: "Count Update Done!",
                    Data: 1,
                    IsSuccess: true
                });
            }
            else {
                res.status(400).json({
                    Message: "Data not found!",
                    Data: 0,
                    IsSuccess: true
                });
            }
        }
        else {
            res.status(400).json({
                Message: "Invalid User Credentials!",
                Data: 0,
                IsSuccess: true
            });
        }
```



```
    } catch (err) {
        res.status(500).json({
            Message: err.message,
            Data: 0,
            IsSuccess: false
        });
    }
}
```

OUTPUT SCREENSHOTS:

Login Screen:



Welcome Back!

keshavgarg@gmail.com

••••••

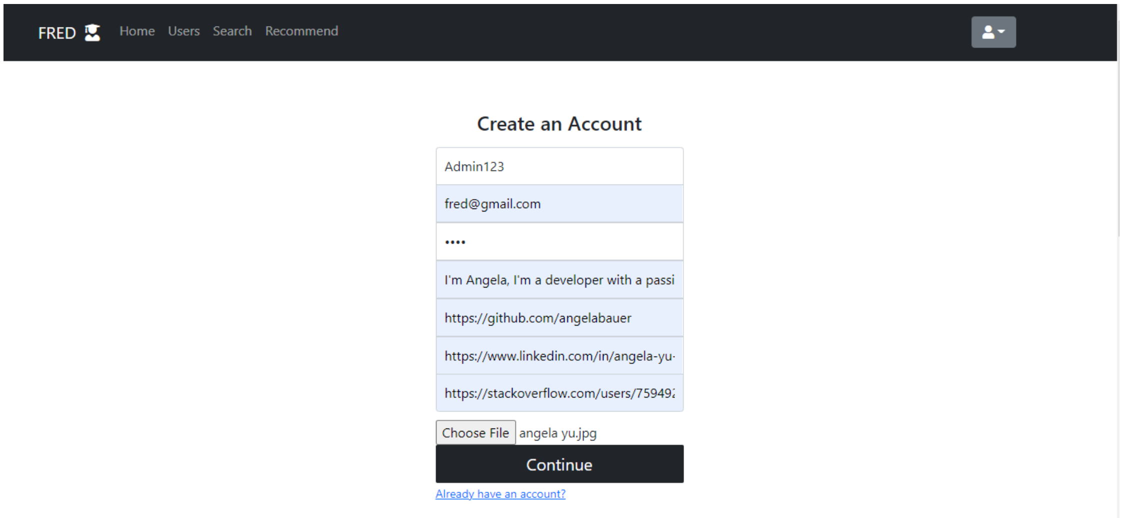
Need an account? Register

© Fred

Join the newsletter to receive our all the updates

You can unsubscribe at any time.

Signup Screen:



Create an Account

Admin123

fred@gmail.com

••••

I'm Angela, I'm a developer with a passi

https://github.com/angelabauer

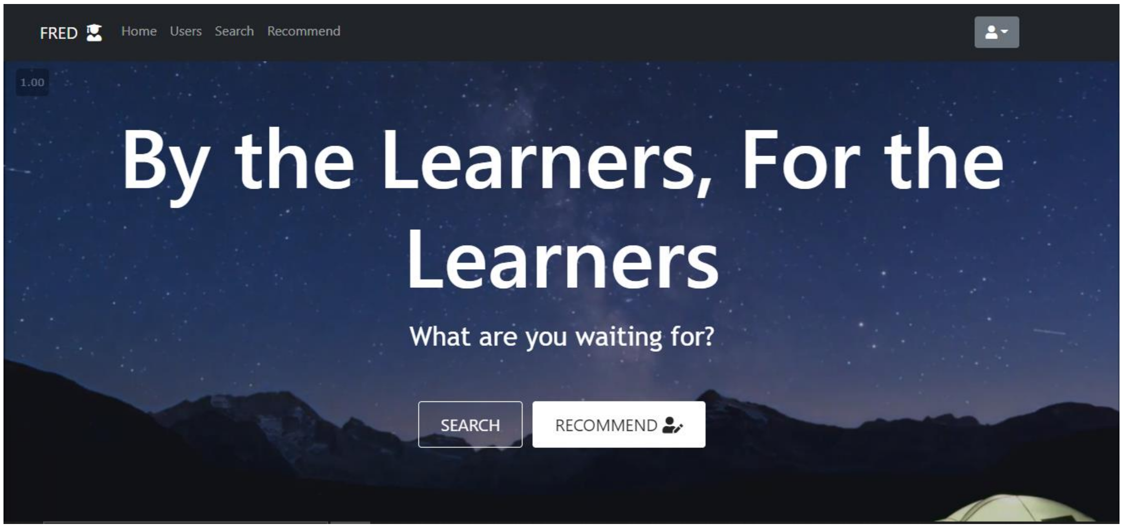
https://www.linkedin.com/in/angela-yu-

https://stackoverflow.com/users/75949;

Choose File angela yu.jpg

Already have an account?

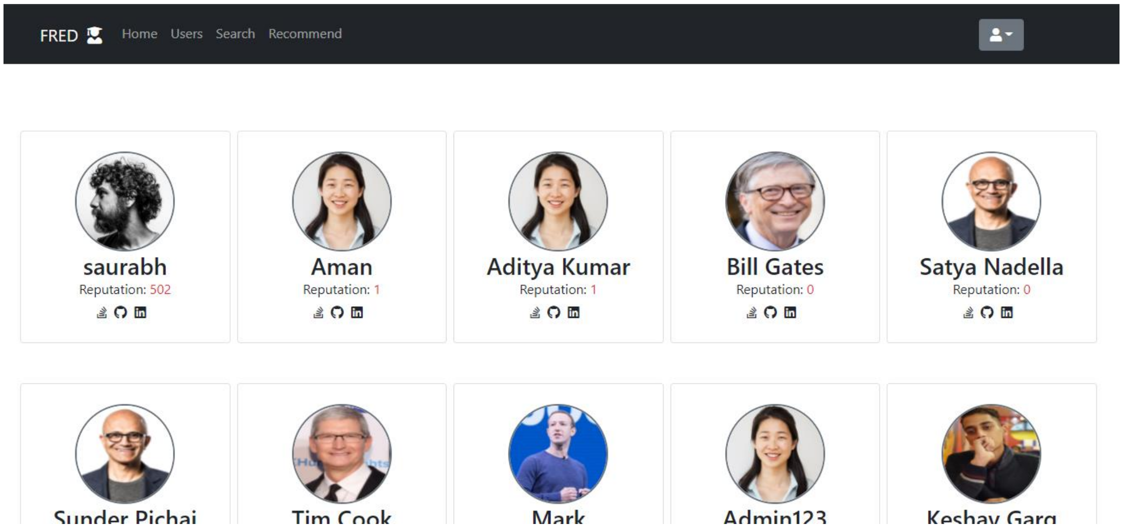
Home Page:



By the Learners, For the Learners

What are you waiting for?

Users List: Here you can add credibility i.e. reputation to your favourite users based on their skills.



saurabh
Reputation: 502

Aman
Reputation: 1

Aditya Kumar
Reputation: 1

Bill Gates
Reputation: 0

Satya Nadella
Reputation: 0

Sunder Pichai

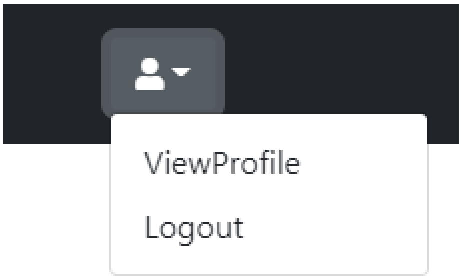
Tim Cook

Mark

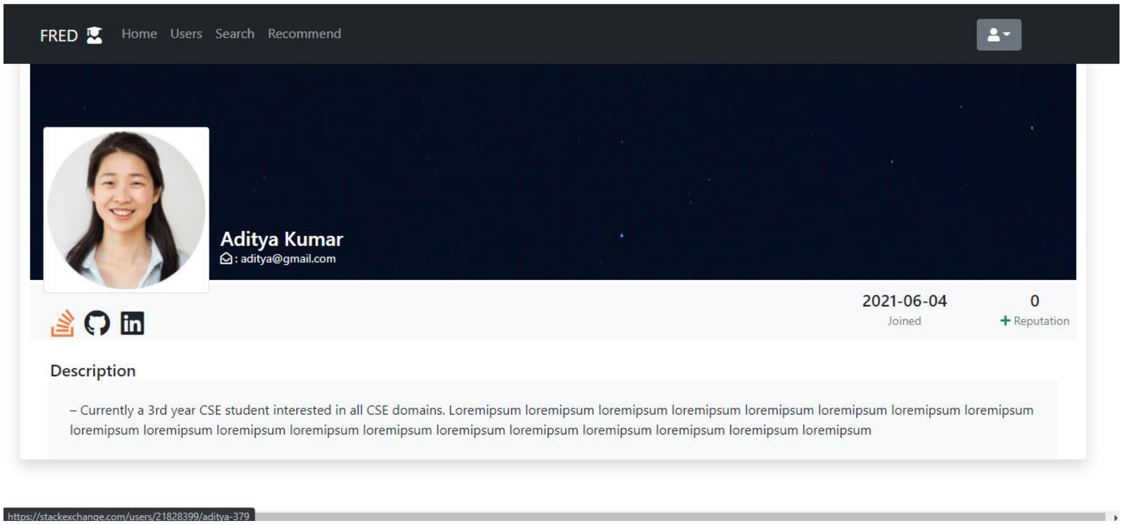
Admin123

Keshav Garg

View Profile and Logout Option in Dropdown:



User Profile Page: Here you can mail the user, look at their description, LinkedIn, StackOverlfow and GitHub account to see how credible they are and based on your understanding you can add ‘1’ reputation to the user.



Aditya Kumar
: aditya@gmail.com

2021-06-04
Joined

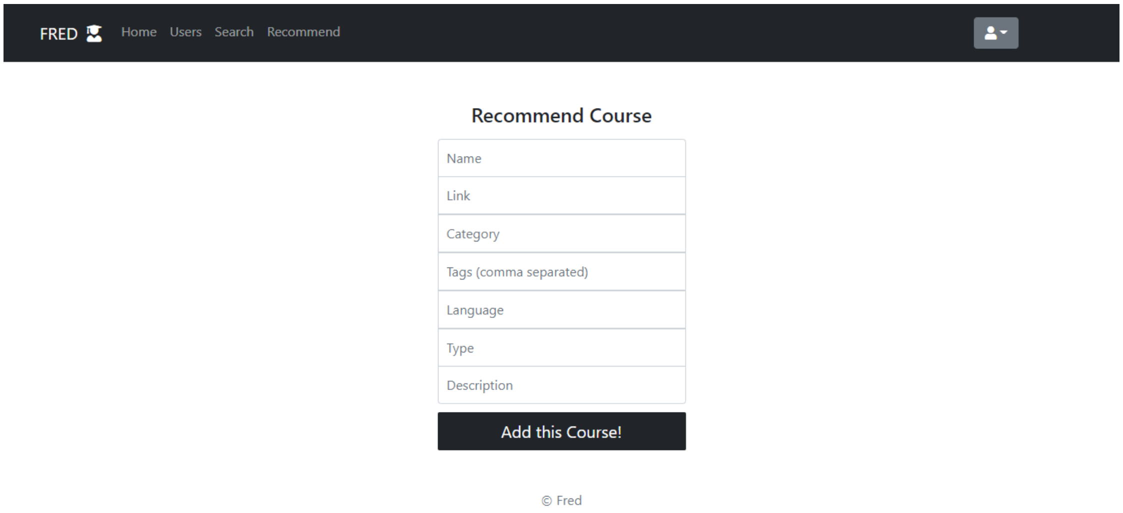
0
+ Reputation

Description

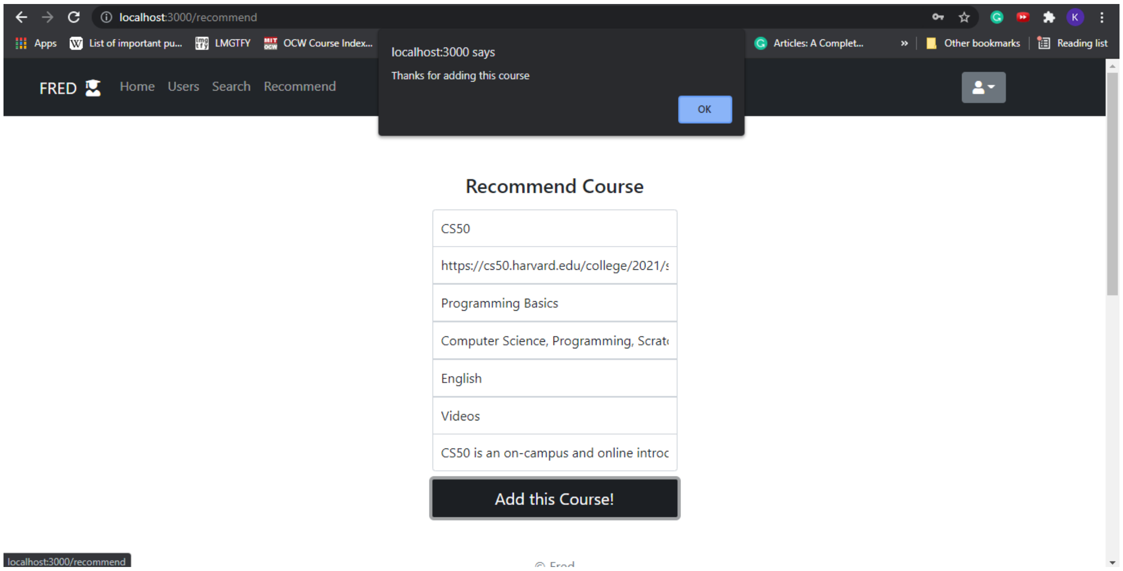
– Currently a 3rd year CSE student interested in all CSE domains. Loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum loremipsum

https://stackexchange.com/users/21828399/aditya-379

Recommend Page: Here you can recommend any free course, videos, blogs that you consider to be of value to others



After Successful addition it will tell you that the course is added successfully



localhost:3000 says

Thanks for adding this course

CS50

https://cs50.harvard.edu/college/2021/s

Programming Basics

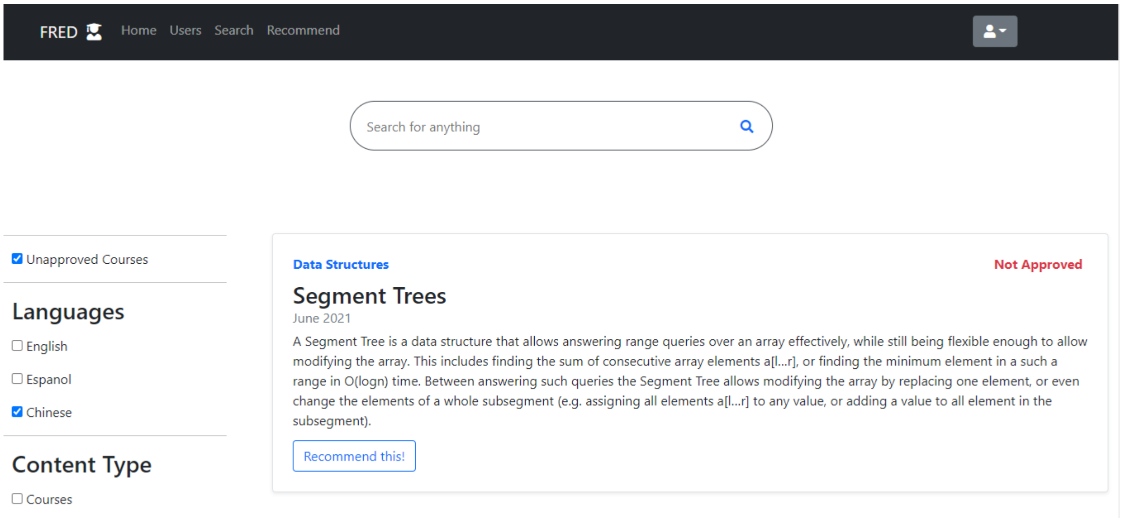
Computer Science, Programming, Scratc

English

Videos

CS50 is an on-campus and online introc

Search Page: Sorting Unapproved courses on the basis of language



☑ Unapproved Courses

Languages

☐ English

☐ Espanol

☑ Chinese

Content Type

☐ Courses

Data Structures — Not Approved

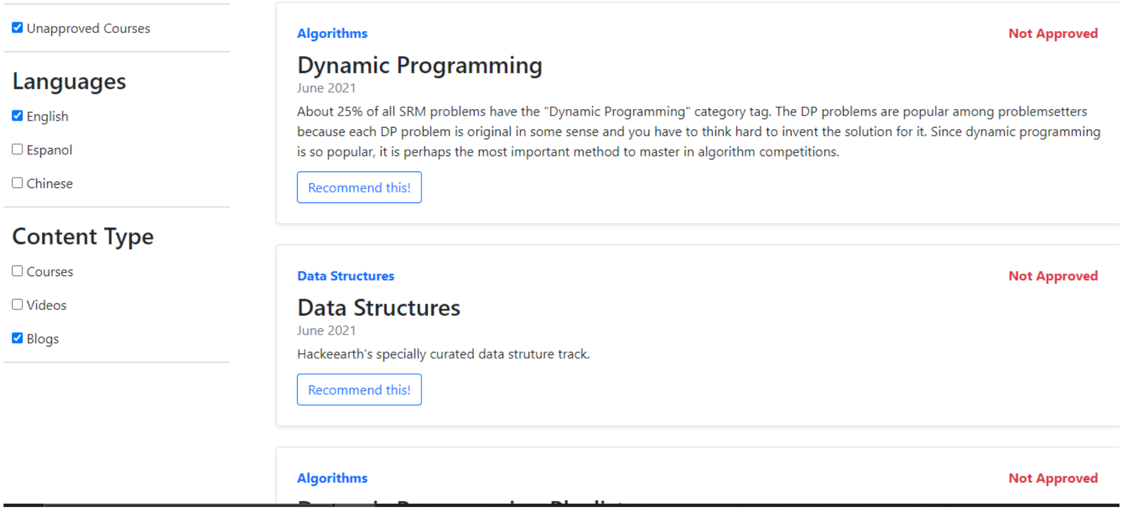
Segment Trees

June 2021

A Segment Tree is a data structure that allows answering range queries over an array effectively, while still being flexible enough to allow modifying the array. This includes finding the sum of consecutive array elements a[l...r], or finding the minimum element in a such a range in O(logn) time. Between answering such queries the Segment Tree allows modifying the array by replacing one element, or even change the elements of a whole subsegment (e.g. assigning all elements a[l...r] to any value, or adding a value to all element in the subsegment).

This courses are in unapproved list. If you like this course you can click on Recommend this button. Your user reputation or 500 whichever is smaller will be added to the course reputation and if it crosses a certain threshold, 5000 to be precise, it will come in the approved course list. The “Not Approved” tag in the top right corner will also disappear. Admins can add 5000 reputation.

Sorting unapproved courses on the basis of language and content type



☑ Unapproved Courses

Languages

☑ English

☐ Espanol

☐ Chinese

Content Type

☐ Courses

☐ Videos

☑ Blogs

Algorithms — Not Approved

Dynamic Programming

June 2021

About 25% of all SRM problems have the "Dynamic Programming" category tag. The DP problems are popular among problemsetters because each DP problem is original in some sense and you have to think hard to invent the solution for it. Since dynamic programming is so popular, it is perhaps the most important method to master in algorithm competitions.

Data Structures — Not Approved

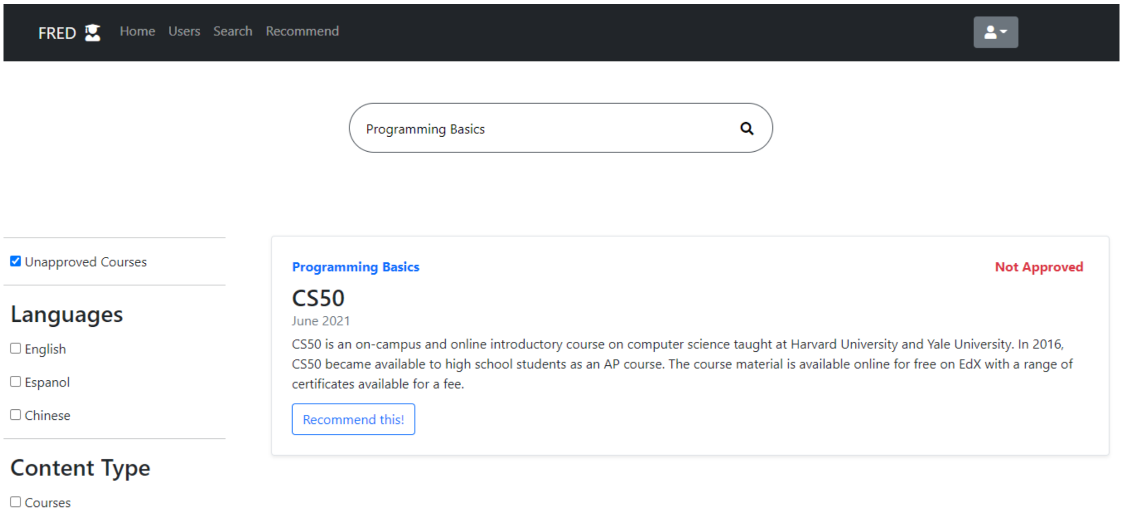
Data Structures

June 2021

Hackeearth's specially curated data struture track.

Algorithms — Not Approved

Search Page: Sorting unapproved courses on the basis of search query



Programming Basics

☑ Unapproved Courses

Languages

□ English

□ Espanol

□ Chinese

Content Type

□ Courses

Programming Basics

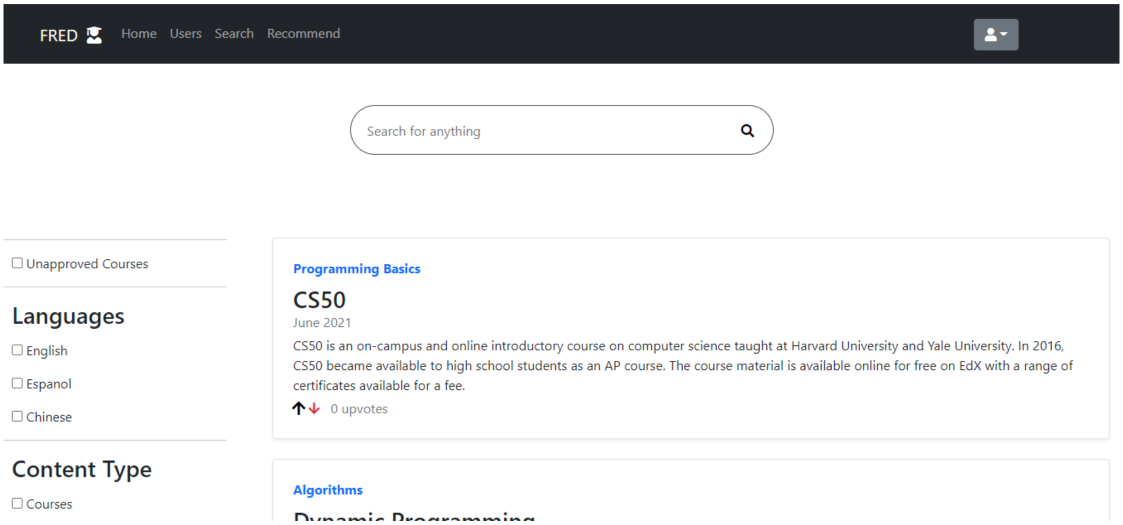
Not Approved

CS50

June 2021

CS50 is an on-campus and online introductory course on computer science taught at Harvard University and Yale University. In 2016, CS50 became available to high school students as an AP course. The course material is available online for free on EdX with a range of certificates available for a fee.

Approved Course List



□ Unapproved Courses

Languages

□ English

□ Espanol

□ Chinese

Content Type

□ Courses

Programming Basics

CS50

June 2021

CS50 is an on-campus and online introductory course on computer science taught at Harvard University and Yale University. In 2016, CS50 became available to high school students as an AP course. The course material is available online for free on EdX with a range of certificates available for a fee.

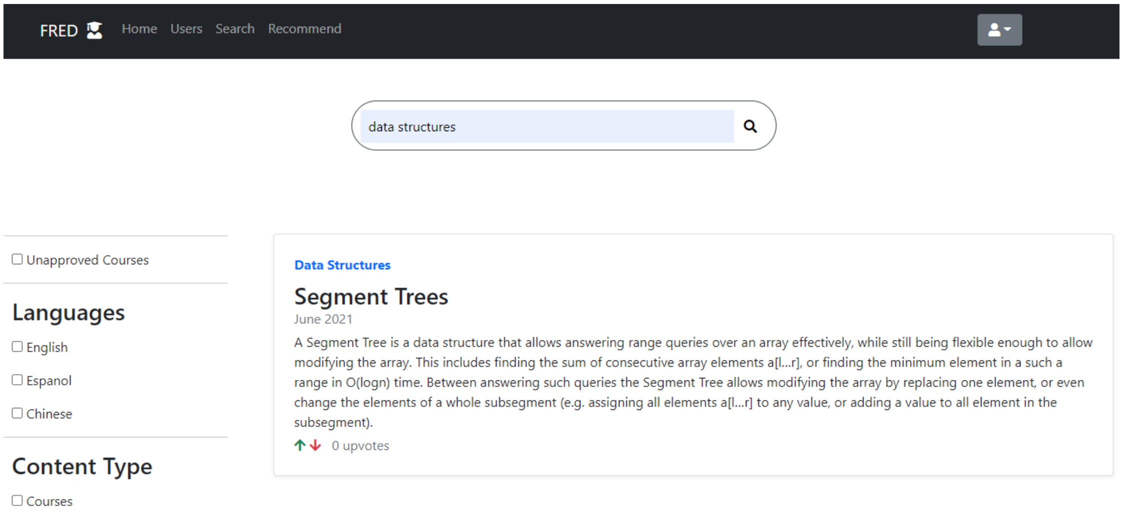
↑↓ 0 upvotes

Algorithms

Dynamic Programming

Note: I have added all the courses in approved list by increasing their reputation

Search for approved courses by query, here we can upvote or downvote the course one time per user



data structures

☐ Unapproved Courses

Languages

☐ English

☐ Espanol

☐ Chinese

Content Type

☐ Courses

Data Structures

Segment Trees

June 2021

A Segment Tree is a data structure that allows answering range queries over an array effectively, while still being flexible enough to allow modifying the array. This includes finding the sum of consecutive array elements a[l...r], or finding the minimum element in a such a range in O(logn) time. Between answering such queries the Segment Tree allows modifying the array by replacing one element, or even change the elements of a whole subsegment (e.g. assigning all elements a[l...r] to any value, or adding a value to all element in the subsegment).

↑↓ 0 upvotes

Courses are sorted based on upvote-downvote ratio