Search Engine Optimized Content Sharing Platform – Chaldea

Project Report

Moukhik Misra

ABSTRACT

Search Engine Optimization refers to the process of improving the visibility of your particular content or website. The Internet is a gigantic place with many different website containing subjects related to all sorts of different topics. Hence it is important to have a system in place that allows users to search their content as per their preference and to show them related searches as per their search to topics that may interest them or to guide them to relevant articles, sties or blogs which can give them more information related to their initial search. This is known as Search Engine Optimization (SEO). The main objectives of SEO are to make sure that users are able to see search results related to their original search which can possibly help them decrease their time searching for appropriate content as well as helping the owners of the website or content generate more traffic towards their website which will help improve their visibility, popularity and business. In our project we have implemented a small-scale version of Search Engine Optimization which is being utilized on a content sharing platform called Chaldea. Chaldea is a text and image sharing platform where users can write articles as per their choice on topics of their choosing and can add images to their content and share it to users all over the platform. Chaldea allows users multiple functionalities such as creating blogs, adding pictures to the blogs, setting personalized styles to their blogs as per their choosing. The implementation of SEO is done by means of categories and tags which are the main driver for the related content search. The categories and tags can be set by the user while they are creating their blogs and based on the set categories and tags, their content will be displayed to other users on basis of their searches and related blogs. When users will go to a particular blog, they

can find that at the bottom of the page, they will be linked to related blog posts made by other users as determined by the SEO. The platform also comes with Admin functionalities where admins have the power to update or delete any of the blogs as well as create, update and delete the tags that can be used by the general user. The platform is extremely scalable which implies that it can be used on a large scale when deployed and it uses protected rotes and appropriate middleware to make sure that every user or admin gets directed to the page that they are actually supposed to go to and not to any random pages. The data handled by the platform is also secure and usernames are hashed and hence securely stored in the database leads to a data being securely stored.

INTRODUCTION

SEO stands for search engine optimization, which is a set of practices designed to improve the appearance and positioning of web pages in organic search results. Because organic search is the most prominent way for people to discover and access online content, a good SEO strategy is essential for improving the quality and quantity of traffic to the owner content or website. Search engine optimization is a key part of online marketing because search is one of the primary ways that users navigate the web. Search results are presented in an ordered list, and the higher up on that list a site can get, the more traffic the site will tend to receive. For example, for a typical search query, the number one result will receive 40-60% of the total traffic for that query, with the number two and three results receiving significantly less traffic. Only 2-3% of searchers click beyond the first page of search results. Thus, even a small improvement in search engine rankings can result in a website or content receiving more traffic and potentially business. Because of this, many businesses and website owners will try to manipulate the search results so that their site shows up higher on the search results page (SERP) than their competitors. This is where the search engine optimization becomes extremely essential for both users and owners for display of content and website. In our application Chaldea, SEO is implemented using categories and tags which are assigned to blogs when they are created first. These categories and tags can be updated by the user later down the line as per their needs. When the user creates a blog with particular categories and tags, it is then used by the SEO module to display that particular content to users viewing other blogs as well.

Chaldea allows its users to create blogs in multiple ways and it provides various functionalities to the user for creating their required blogs. Users can also add a poster image to their blogs if they so choose to do so. The application also comes with a lot of other user functionalities such as a universal blog page where users can view the latest created content form all the other users on the platform. It also allows the users to create, update and delete their blogs as per their requirement. Another functionality is that users are able to create their own public profile for others to view. A blog is associated with a particular user and the users are able to view the public profile of the blogs created by that particular author as well as get links to all the other blogs created by that user. It is an extremely useful method to let users search fo logs by their favourite authors.

In our application, we use secure protected routes for letting users go to pages that are only meant to be visible to them. Protected Routes are routes that can only be accessed if a condition is met(usually, if user is properly authenticated). It returns a Route that either renders a component or redirects a user to another route based on a set condition. In our application protected routes have been implemented by use of token system which is generated on basis of user and role of the user. Now if user is an admin then they are directed to the admin functionalities page, and if they are users they are directed to the user page as per the necessary role. Also, by means of protected routes, we ca make sure that both users and admins are allowed functionalities on the bais of their role and that no functionality is mixed up. Protected routes allow users to ensure only logged in users can access certain parts of the site that may contain private user information. Users can also be assured that all their sensitive data such as username and password are hashed and stored in the database meaning that there can be no link made to the given real life person and the account created to the data that is being stored in the database.

AIM

Given below are the aims that we set out to achieve while planning the project:

- To create a content sharing platform that enables users to create blogs, add pictures and display it to all users on the platform.
- To have a login and signup module for adding users.
- To create a basic Search Engine Optimization framework usable within the website that will display the most relevant blogs created by users to people searching for relevant blogs. This will be done by means of categories and tags.
- To provide multiple functionalities to the user for creating their blogs and to allow different text formatting to be use while creating said blogs.
- To enable users to create, update, and delete their blogs and update the categories and tags as per their requirement.
- To have admin functionalities where the admin can create categories and tags which can subsequently be used by users in their blogs. Admins should also be able to remove these categories and tags. Admins also should have the authority to delete and update blogs and can create their own blogs too.
- To have protected routes implemented by means of a token generated based on user and role (normal user or admin) and to make sure that each user is directed to the correct pages and that they have access to functionalities authorized to them.
- To make sure that the data in the database is secure by hashing the sensitive data stored in the database.

OBJECTIVES

Give below are the objectives of our project:

- To implement create blogs functionalities by using react quill which is the Quill Rich text editor. This will provide users to opportunity to have various text formatting abilities for their blogs.
- To have a login and signup module for adding users.
- To allows users to have a descriptive or a cover picture for their blogs which can be displayed on the blog hub.
- To allow users to set their categories and tags as per their choice depending on what category and tags they want their blogs to show up as related searches.
- To enable users to update and delete their blogs as per their choice.
- To allows users to have a public profile page which can have a profile image and this page will be publicly available to all users who want to see the other blogs by this particular user.
- To have a central blog hub where all the latest created blogs can be viewed.
- To have admin section.
- The admins should have the functionality of creating and deleting categories and tags that is to be used by the users while creating their blogs.
- The admins should also have the ability to update and delete all blogs on the platform as well as create their own blogs.
- To implement protected routes and have the necessary Admin and Authentication middleware in place to make sure that the users are being directed to pages only they are authorized to based on role and that no user should be able to access pages beyond their role. This implementation is achieved by using a token system system based on user and role and by implementing cookies and local storage for storing this data.
- To make sure that a hashing function is present which hashes sensitive information which is stored in the database to improve security.

LITERATURE SURVEY

1. Optimized Technique for Ranking Webpage on Search Engine Optimization

Search Engine Optimization refers to improving the traffic on a certain webpage by improving the visibility of the page to the search engine so that it is more visited by users visiting a page. It is important to make sure that the user is seeing content which closely resembles their search keywords and that the most relevant and popular details or web pages are displayed to the user first. Research on SEO shows that 95% of the people use the first 3 links that appear on their search engine results. In the given paper, the authors propose a more optimized method for search engine optimization by proposing a page ranking methodology. In this work, the users' query log is used to understand the rank optimization algorithm and to generate the search results that closely match or are relevant to the search keywords. This can be used to form clusters of page results that can be sorted on the basis of relevance again by using the proposed methodology and this results in an optimal technique for ranking webpages on SEO. the general formula for dynamic rank updation that is followed is new rank= old rank + weight where the weight of every webpage is found from the user query log using adjacency and clustering techniques.

2. The Application of Search Engine Optimization for Internet Marketing: An Example of the Motel Websites

In this paper the author conduct research and surveys on the impact of the application of search engine optimization on the popularity of a case business which in this case is taken as motel website. Several methods of SEO can be applied to ensure that there is increased traffic to the motel website, but it is important for the owners to apply the right one for the correct amount of visibility and reach. Some of the basic SEO techniques implemented here are to modify the HTML and Title label, to contain meta content which can be used to describe the particular webpage to the search engine for better visibility. Subtle techniques such as adding notes to an image also help to boost reach. To ensure that the correct heading size is followed for headings and subheadings. Registering in open website catalogs in the public domain also may attract traffic to the website. To ping the search engines irregularly to notify that the website has been updated. and to post on famous discussion boards and put keywords in in the signature lines. These methods were applied over a period of time following which the results of the SEO techniques implemented were evaluated. The results indicate that the visibility of the website had increased dramatically and it resulted in large gains for the motel owners. Hence form this paper, the author signifies the importance of SEO in internet marketing.

3. Search Engine Optimization to Enhance User Interaction

In this paper, the authors try to highlight the importance of SEO techniques on the user experience. SEO is widely considered to be one of the best marketing techniques available for vendors or owners of business. However, one must also consider the benefits of SEO on user experience. Through the use of SEO techniques, users can quite easily get to the relevant content that they were searching for on the web or they may be prompted to relevant content on the Internet via ads or other methods. The SEO process involves research namely keyword analysis and competitive analysis. This is followed by reporting and goal setting for websites which is an important metric for measuring the traffic and activity of a website. this is followed by the content building phase which provides better competitive environment in SEO. Following which is the webpage optimization process which are related to the GUI of the website. The appearance of media components is extremely important. he most important being webpage titles, content exploration as in categories and tags, prominence of targeted keyword phrases and the site outline. Lastly an important step is social and link building for measuring and increasing reach. All these techniques are extremely useful for both the end user as well as the owners of the website

4. A Study on Tactics for Corporate Website Development Aiming at Search Engine Optimization

In this paper, the authors analyse the impact of search engine optimization in context of large scale corporate web development. The authors aim to get an understanding of SEO algorithms commonly used and propose several optimization techniques that can be used while developing a website. The paper goes through multiple established SEO algorithms such as the Page Rank Algorithm, the HillTop Algorithm as well as newer developments in the field of search engine optimization. The authors also propose several tips and tactic for website content, tactics for keywords, tactics for links. The authors say conclude that the search engine spiders can only determine the quality of a website based on its content rather than flashy features or images. Enough content should be provided so as to enable the search engine to index the page as something worthwhile or something with weight. other than this the authors also conclude that keywords and links are also extremely important for page visibility and content accessibility

5. Study on Website Search Engine Optimization

In this paper the authors conduct a broad study on website search engine optimization. With the fast-growing internet, SEO is becoming extremely important for gaining any relevance or visibility on the internet. By utilizing SEO techniques, it is possible to improve the visibility of our content by a significant amount and generate traffic on our website which will help potential owners of businesses by increasing their scale. Many factors affect search page ranking such as webpage correlation, link weights and other time based factors. Keyword matching is an extremely important step for search engine optimization as well as presence of links on website to improv overall link weight. Keyword Optimization depends on related content and large quantity of searches with less competition. It is important for the keyword to be specific and not cover too many topics as it dilutes the search and decreases the overall weight of search. Keyword density and Distribution are also essential factors for SEO. Apart from this, content optimization, structure optimization and link optimization are all techniques to be used for SEO. The website search engine optimization is a complete series of optimization processes. From the layout of the website structure sections, to the specific content designs of webpage title, keywords, domain names, links, etc., all need to consider the performance with the search engine. In addition, to complete a website developers need to continue to study the characteristics of their own websites and research effective SEO strategy, as well as constantly update the website content to increase traffic, thereby enhance the market competitiveness of the website.

TECHNOLOGIES

1. Next.js

Next.js is an open-source development framework built on top of Node.js enabling React based web applications functionalities such as server-side rendering and generating static websites. Next.js is a React framework that enables several extra features, including server-side rendering and generating static websites. React is a JavaScript library that is traditionally used to build web applications rendered in the client's browser with JavaScript. Developers recognize several problems with this strategy however, such as not catering to users who do not have access to JavaScript or have disabled it, potential security issues, significantly extended page loading times, and it can harm the site's overall search engine optimization. Frameworks such as Next.js sidestep these problems by allowing some or all of the website to be rendered on the server-side before being sent to the client. Next.js is one of the most popular components available in React. It is one of several recommended "toolchains" available when starting a new app, all of which provide a layer of abstraction to aid in common tasks. Next.js requires Node.js and can be initialized using Node Package Manager.

2. Express js

Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js. Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile application. With a myriad of HTTP utility methods and middleware at your disposal, creating a robust API is quick and easy. Express provides a thin layer of fundamental web application features, without obscuring Node.js features.

3. Middleware

i) Morgan

Morgan is an HTTP request level Middleware. It is a great tool that logs the requests along with some other information depending upon its configuration and the pre-set used. It proves to be very helpful while debugging and also if users want to create Log files.

ii) cookieParser

A cookie is a piece of data that is sent to the client-side with a request and is stored on the client-side itself by the Web Browser the user is currently using. With the help of cookies, it is easy for websites to remember the user’s information, it is easy to capture the user’s browsing history, it is also useful in storing the user’s sessions. The session makes requests to all the servers using a secret Id. The information is stored on the server that is linked to this secret ID. To make use of cookies in our application, cookie-parser middleware is used.

4. CORS

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options. CORS is shorthand for Cross-Origin Resource Sharing. It is a mechanism to allow or restrict requested resources on a web server depend on where the HTTP request was initiated. This policy is used to secure a certain web server from access by other website or domain. For example, only the allowed domains will be able to access hosted files in a server such as a stylesheet, image, or a script.

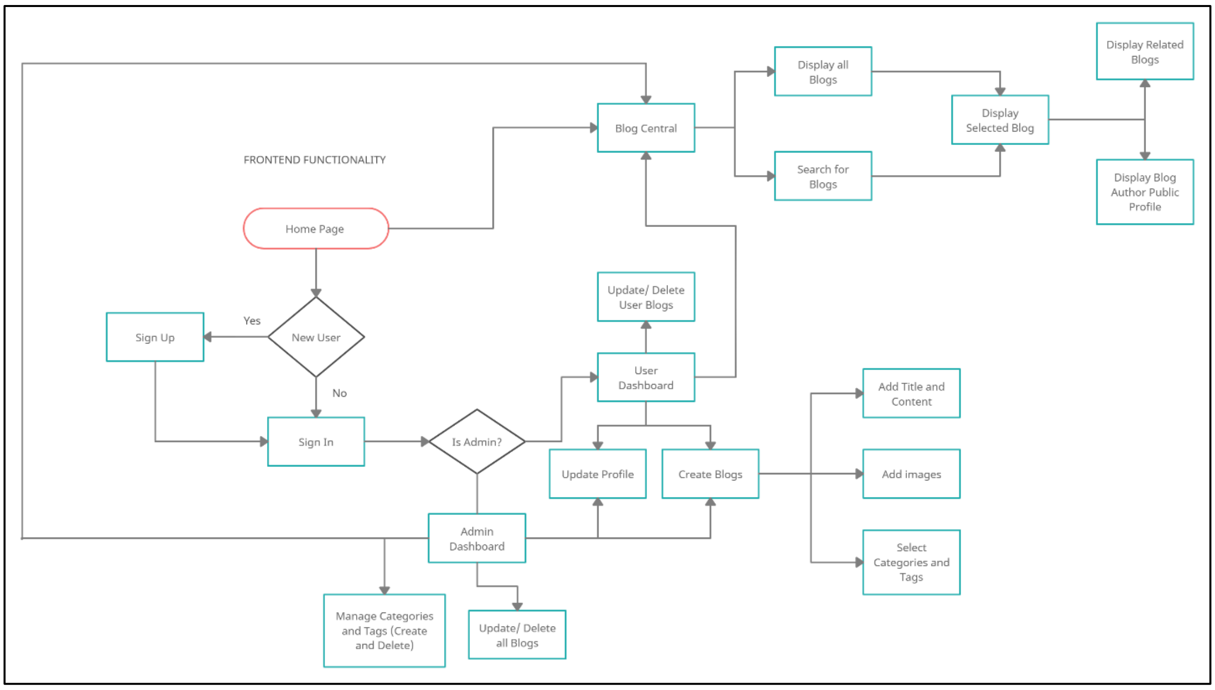
5. Node.js

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts. Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications. Node.js supports JavaScript natively.

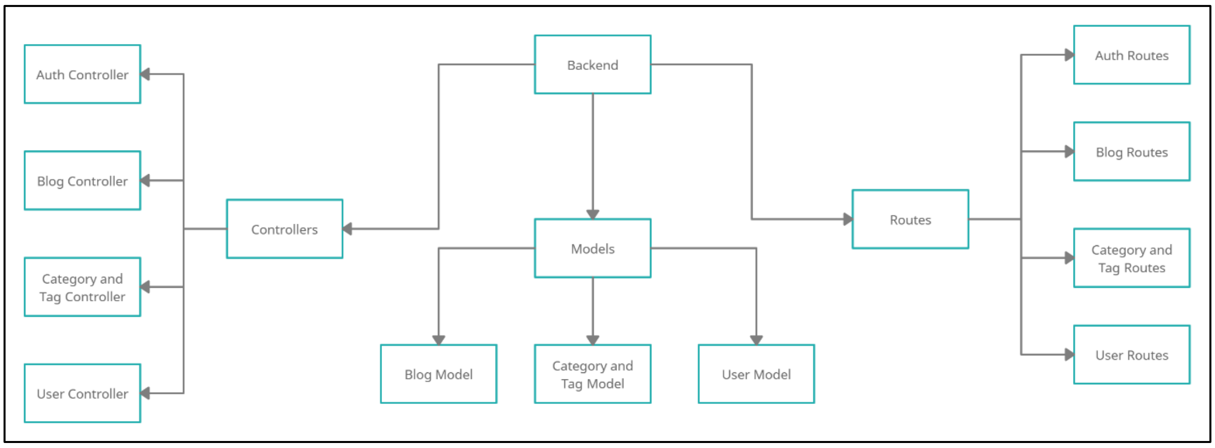
6. MongoDB

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL). In a NoSQL database, data is stored in the form of a document. Mongo DB is JSON like structure database which provide us with the facility to use JSON objects to store data in database. It provides high performance, high availability and automatic scaling.

FLOWCHARTS



Frontend Functionalities



Backend Modules

IMPLEMENTATION

1. BACKEND

a) Main server.js file

In the server.js file, we import the necessary node models that are required for the application. The require statements that we need are to express, morgan, body-parser, cookie-parser, cors, mongoose, and a dotenv.config(). Next in the server file, we bring in the routes into one file which will be described later in the implementation. The routes that we need to bring in are as follows. The routes that are created in the project are the blog route, the auth route, the user route, the category route and the tag route. Next we need to initialize express by using express(). Then, we define the connectivity to the database using mongoose.connect() and then on successful connection, we can pass a success message in the console. Next we need to add the middlewares that we are using in the application. The middlewares that we are using in the project are morgan, express.json and cookieParser. Next we need to initialize the cors. When the application is in production or development mode, we can use cors. The functionality of cors is defined in the technologies section above. Then, we can set the routes middleware for the blog routes, the auth routes, the user routes, category and tag routes. This route middleware adds the /api url to the necessary application sites. Finally in the server.js file, we define the port for the server. The server port is set to 8000.

b) ROUTES

i) auth Route

The first route that is created is the auth route. This route is responsible for directing to the signup, signin, signout and the requireSignin sections which will be explained in the controllers section of the backend. These routes will also implements the validators called userSignup Validator and the sign in validator which are defined in the validator section of the backend. Next we add the necessary route to the site which are namely /signup for the signup which is defined as a post method and we pass necessary objects too. For signin the site extension is signin as a post method with eh necessary validators, and run validation for signin. Finally for signout the extension is /signout which is defined as a get method.

ii) blog Route

The blog.js route requires the create blog, list blog, list all blogs with categories and tags, read blogs, remove blogs, update, photo, list related blogs, search and list by user which are all defined in the blog controller backend. Next the necessary valiators for require signin, admin and auth Middleware and the can update and delete blog section are required from the auth controller. Next the necessary post, get and delete methods are defined. First method is the create blog post method. Next is the List blogs get method. They have extension /blog and /blogs respectively. The next one is post method to display all the blogs with categroies and tags. The next is the read bog via its slug value. The next delete method is to delete blog followed by the update blog put method followed by a get photo route. This is then followed by the post method responsible for displaying the related blogs and then the get method defied for Search functionality. Next routes are defined for authenticated users where they can create blogs, list blogs by user and delete blogs as a delete method. Finally authenticated user can also update their blogs via put method. All the necessary routs have their respective admin or auth Middlewares as per the functionality for admin or user. Finally we export them to the router.

iii) Category and tag routes

Both categories and tag routes have similar make up and functionality. They both require category and tag controller with the create ,list, read and remove functionality. Then we need the necessary validators and run the validation, create category and tags validator and the require signin and the admin Middle ware validator . The create category an tags functionality is available only to the admin hence the need for the admin Middlewre for the protected route to only allow users to this page. Next for both categories and tages we have a post method for creating the category or tag followed by a get mthod to list all the caategories and tags and then another get method to read the tags and finally a delete method to delete the category and tags.

iv) User route

The final route is the user route which requires the user and the auth controller. This route is primarily to direct the users to their profile page whether it be an admin or a regular user. For this route we require the signin and the auth and admin middlewares from the auth controller followed by the read, public profile, update and photo form the user controller. The method defined are the get method to read the user profile followed by a get method to display user profile. followed by the update user profile route get method and finally the user profile picture route which are all exported finally.

C) CONTROLLERS

i) Auth controller

The auth controller requires the user model defined later and the blog model and the short Id and json web token and express jwt. The first export is the read which returns the profile of the requested username. The next export is the signup export where we need to check if ithe email is already taken or not. The application needs unique email ids for different users. If email is already taken, send back error message. This export also generates the short ID for the username and to define the profile page for the created user and to save the inputted details in the fields provide to the database. If the details have been successfully saved, we can pass the success message saying that signup successful. The next export that is deined is the signin export which is used to check if the user exists. User needs to enter email id and password. We need to send error if email does not match to any in the database. Next we authenticate the user with password and provide necessary error message if required. Net we generate the jwt token for the middlewares and we send it to the client. We set a cookie with token which expires in 1 day. Finally we can send all these details as a response json wih token and user details to be used later. The next export is the sigout export which clears the cookie of the token for signout. This is followed by the require signin export using JWT with algorithm set as HS256 with appropriate user property. The next export is the auth Middleare to authenticate the use basically to find the user by ID and to return necessary success or error message. The next export is the admin Middleware which is usd to check if the user signing in is actually an admin or not. If the user role is 1 in the database, the the user is considered to be an admin. If the user role is 0 then it is normal user and the admin resources are not available to them. Finally the last export is the can Update and delete blog which is used to update and delete blogs for the users and admins depending upon the slug of a blog. A slug is an identifier for a particular blog and can be thought of as the blog id. We also check the authorization status of the user to determine whether they are authorized or not to create and update and delete blogs.

ii) blog controller

The next controller is the blog controller which requires the blog, category, tag, user models It also requires formidable and slugify and string-strip-html and lodash and the errorHandler defined below finally we require the smart trim function defined in the blog helper. The blog controller is used to create a blog and check if everything is updated properly or not. We can send error message if anything does not upload properly. We also heck the title, body, categories, tags. The title field cannot be empty. The blog body needs to have minimum of 10 characters. Atleast one category and tag needs to be selected. Then we apply the Blog(0 function and set the blog title, blog body and smart trim to get an exceprt that can be displayed on the central hub page, then we can generate blog id by slugify and set a meta title necessary for SEO and a meta description that will be a stripHTML of the body of 160 characters maximum. Then the posted by variable is also defined. The categories and tags are also placed in the necessary arrays. Photos can also be uploaded with size less than 1 mb. Finally the blog can be saved and success or error meassage sent. The next function is to find the blog y id and to update the blog. The subsequent exports invlove exporting the list of all blogs created as well as a separate export with as list all the blogs with categories and tags. Next fnction s the blog find function via he categories and tags. We can also have a functionality here if we click a particular category or tag, all blogs corresponding to the corresponding categories and tags are displayed. The next export involves reading the details of the blogs created and to contain all the data of the blog present in the database. The next export involves removing the said blog if needed and to pass the required success or error message. The next export is to update the blog. In blog update users can change allt he details for their blogs including the title, categories, tags, body and photo as per their requirement. The next export delas with the upload of the photo which is present in the blog or the blog cover photo. The then next export deal with the related blogs that are obtained via the SEO. The next export delas with the search functionality where users can search for blogs based on keywords present. Lastly there is an export which deals with the list the blogs created by a user on their public profile page.

iii) Category and tag controllers

The category and tag controllers work in similar ways where both require ther category of the tag model respectively, they require the blog model defined , slugify and the error handler. First export is the category and tag creation where the admin can create categories or tags which can be used b users. The next export is the list export where all categories and tags are listed out. This si followed by the read export which finds th categories and tags associated with a particular blog. The last export is the remove export by which the admin can remove categories and tags.

iv) User Controller

The user controller requires the user model and the block model as well as lodash, formidable, fs and error handler. The first export is the read export which returns the user profile as a response json. The next export is the publicProfile export which I ued to dislay the public profile of the user and it contains the user public details ,profile pcture and all the blogs created byt the user. The next export is the update export and this export is required to define the updation of the user profile such as name, password, profile picture and so on. Latly, there is a photo export which is responsible for importing the profile picture of the user form system.

d) MODELS

i) Blog Model

The blog model defines the blog schema in the database. In our application, the blog schema consist of a title of type String on min length 3 and max length 160 and is a required field. This is followed by the slug string field which is a unique blog identifier. This si followed by blog body of type object and is required with minimum 10 characters and max size of 2 mb. The next field id the excerpt field to be displayed on the blog central hub. This is an trimming of the content of the blog with max 1000 characters. These are followed by the meta title and meta description of each of the blogs and are of type object. The blog schema has a photo field for the cover photo of the blog. The following fields are the categories and tags fiels which are both required and a posted by field to recognize the author of the blog. The timestamps are also enabled to see when the user posted the blog.

ii) Category and Tag models

Both category and tag models are separate yet follow a similar structure. They both have a name field of type string which is required and max size of 32 and they both have slug flied which is the unique identifier of the category or tag created as well as a timestamp.

iii) User Model

The user model consists of the username of type string a required field of max length 32 which is unique and lowercase as well as a name field of type string and required and maxlength 32. This is followed by an email field of type string which required and unique and lowercase. The next field id profile of type string and is required and the field after that is the hashed password field of type string and is required followed by the salt string field, the about field and the role field which si of type number and is default as 0. This is followed by the photo field which is corresponding to the profile picture of the user. This is followed by the timestamps of the creation. Next is a function which creates the hashed password for the given password and gives the salt values. These functions are known as the encypt assword and the make salt functions and they are responsible for the above defined tasks.

e) HELPER FUNCTIONS

i) blog helper function

This function is called the smart trim function and it is used to basically trim the input on both sides if present.

ii) Error Handler helper function

The error handler function deals with the errors that were discussed in the previous sections suchas field already exists error, or any sort of error code sent by the borser on the client side or server side.

f) VALIDATORS

i) auth Validators

The first validator in auth validator is the userSignup Validator whichc ehcks whether the name field is filled, checks for a valid email, checks if the password field is minimum 6 characters long. The next validator expor tin auth Validators is the user Sign In Validator which validates whether the email exists in the database and to check whether password is 6 characters long.

ii) Category and Tag Validator

These validators check whether the category and tag names are empty or not while creating categories and tags.

iii) Index Validator

The index validator is used to run the mentioned validations on the client side and provide th error messages.

2. FRONTEND

a) ACTIONS

i) Auth Actions

In the auth actions, we handle the actions related to handling responses for signin such as the session expired for re sign in. The next action is the signuo action which is defined as a post methods to accept json data and return response.json(). The next action deals with user sign in which is a post method that deals with the sigin route and the json data and the response .json() is sent. The next action is the signout action which remives the set cookie for the user and diplays the successful signout message. The next action defined is the setCookie action which sets the necessary cookie in the browser. This is followed by the remive cookie action which does as the name suggests. Another action is the getCookie action which returns the cookie according to the key passed. This si followed by the setLocalstorage action with its key and value where the value is stored in local storage using key. The next action is the remove Local storage which does as the name suggest and remove keys from the local storage as per passed key. This is followed by the authenticate action which is used to set Cookie and setLoclalstorage if the user is authenticatd. The next action is the isAuth action whichc checks the cookie and local storage for the user to validate whether they are actually the correct user trying to access the data. Anther action is the update User action which is used to updt signed in user.

ii) Blog Actions

The blog actions involve first creating a blog. This is done first by checking whether the user is authenticated and create endpoints as per role and to then allow the user to create their blog. This is a post method based on the bearer token to authenticate the user. The next action is to list all the blogs with their categories and tags. This is a post method and it returns a json response. The next action is the single blog action which is used to display selected blog on the route defined as per blog slug. The next action is the list related action which is used to list the related blogs as per the categories and tags specified and run through the SEO. This is followed by a list all blogs action which is used to display the blogs on a separate web page based on timestamp. his is the central blog hub where all blogs are displayed. The next action is th remove blog option which can be used by users to remove their blogs if necessary. Users can delete all blogs. Similarly another action is the update blog action which is responsible for updating the blog as per the requirements of the user or admin. Finally we have a search action which is used to search blogs with respect to keywords.

iii) Category and Tag Actions

The category and tag actions are similar in that there is a create action which is used to to create categories and tags by the admin for use by users. The next action is the get Categories or the get Tags action which is used to get the categories and tags that have been created. The next action is the single action which is the functionality when a category or tag label is pressed, it brings up all the blogs that contain those particular categories or tags. Finally there exists a remove action to remove the categories or tags.

iv) User Actions

User actions invlolves displaying the user public profile which is a get request that fetches the data in json form and gives json response. The next action is the get profile action which is used o get the profile of the bolg authors public profile which is available to view to everyone. The next action is the update action which is responsible for updating the user profile as per user requirements.

b) COMPONENTS

i) Auth Components

- Admin component is responsible for pushing the signin page if the user is not authenticated and if the user is not admin then to not allow them to access the admin functionalities.
- Private component is responsible for normal signin for the regular user.
- Profile Update component is used to update the user profile and conatis the ecessary information required. It gets the cookie of the user and gets the stored details in the cookie and it initializes the values as well as hold these values when page is refreshed.
- Signin Component is responsible for handling the submission of input data in the fields as well as handle change and the sign In form to type email and password along with signin button
- The signup component is responsible for providing the fields that are necessary for signup such as name, email, password and others and to handle change on these values.

ii) Blog Components

- The first blog component is s the display of the main blog card on the blog central page. Each blog has to displayed in a specific way along with the necessary image and the excerpt of the blog that is given in the body of the blog.
- The next blog component is the search component which is used to search for particular blogs by passing in necessary keywords. The searched blogs will be displayed according to any match with the tile or body with the passed keyword.
- The next blog component is responsible for the display of the related searches in the form of cards after each main blog based on the SEO results.

iii) Crud Components

- The first crud component deals with eh creation of blogs which is done using the react quill module. The react quill module provides various functionalities which can be used to create blogs as per the users liking. It provides many formatting options to the user. This component also deals with he display and setting of categories and tags by the user while creating the blogs and the publishing of the blogs as well. It also can handle page refreshes while maintaining the body of the blog being written in its intermediate state.
- The second crud component is the reading of the blog and this deals with loading of the blog on the site as well as operations for delete and update of the blogs and list out all created blogs by all users.
- The next crud component deals with the updating of blogs. Blogs can be updated by their title, their content, categories, tags images as per user choice and this change is then reflected while next viewing the blog.
- The next crud components are the category and tag components responsible for showing category and tags, creating new ones as well as deleting them by the admin.

iv) Disqus Commenting System

This part imports the widely used multi user commenting API known as DISQUS and we implement it in our program by using the standard functions and code provided in order to integrate it into our application.

v) Header and Layout Component

The header component is responsible for the navbar located on top of the screen with options such as sign out , create blog and it displays the signed in users name. It also provides the first point of redirect when any of these links are clicked and it allows the routing process to begin. The layout component is a very simple representation of the structure of the page.

c) PAGES

i) Admin Pages

- The admin pages consist of several different pages such as the crud pages and the index pages which serves as a landing site when an admin logs in
- The crud pages provide options such as Updating blogs, deleting blogs, creating new blogs, managing blogs and managing categories and tags all in separate pages.
- The index page for the admin contains all the above listed features in tabular format on the lading page with links to the necessary pages.

ii) Blog Pages

- The blog pages are responsible for displaying the blog that has been selected by a user based on the slug value or the identifier of that particular blog. It is responsible for the structure of the blog page and it shows the content and the tags and categories as well as the related blogs and comments.
- The index page handles the blog central hub responsible for displaying all the blogs published to all users. It also displays all categories and tags as well.

iii) Categories and Tags Pages

- The categories and tag pages are responsible for displaying all the blogs that's belong to that category or tag.

iv) Profile Page

- The profile page is responsible for displaying the user public profile as well as all the blogs created by that user based on time of creation.

v) User pages

- The user also has crud pages which let the user update blogs created by them as well as create a new blog and manage their created blogs.
- There is also an update profile page where users can update their public profile and sensitive information such as password and username.
- The user dashboard page contains links to the above-mentioned functionalities as well as to the update profile page.

vi) Index Page

- Index Page is the lading page or the home page that all the people visiting the site get to first see. Its an overview of the entire blog site and provides links to all the blogs and certain specific categories as well.

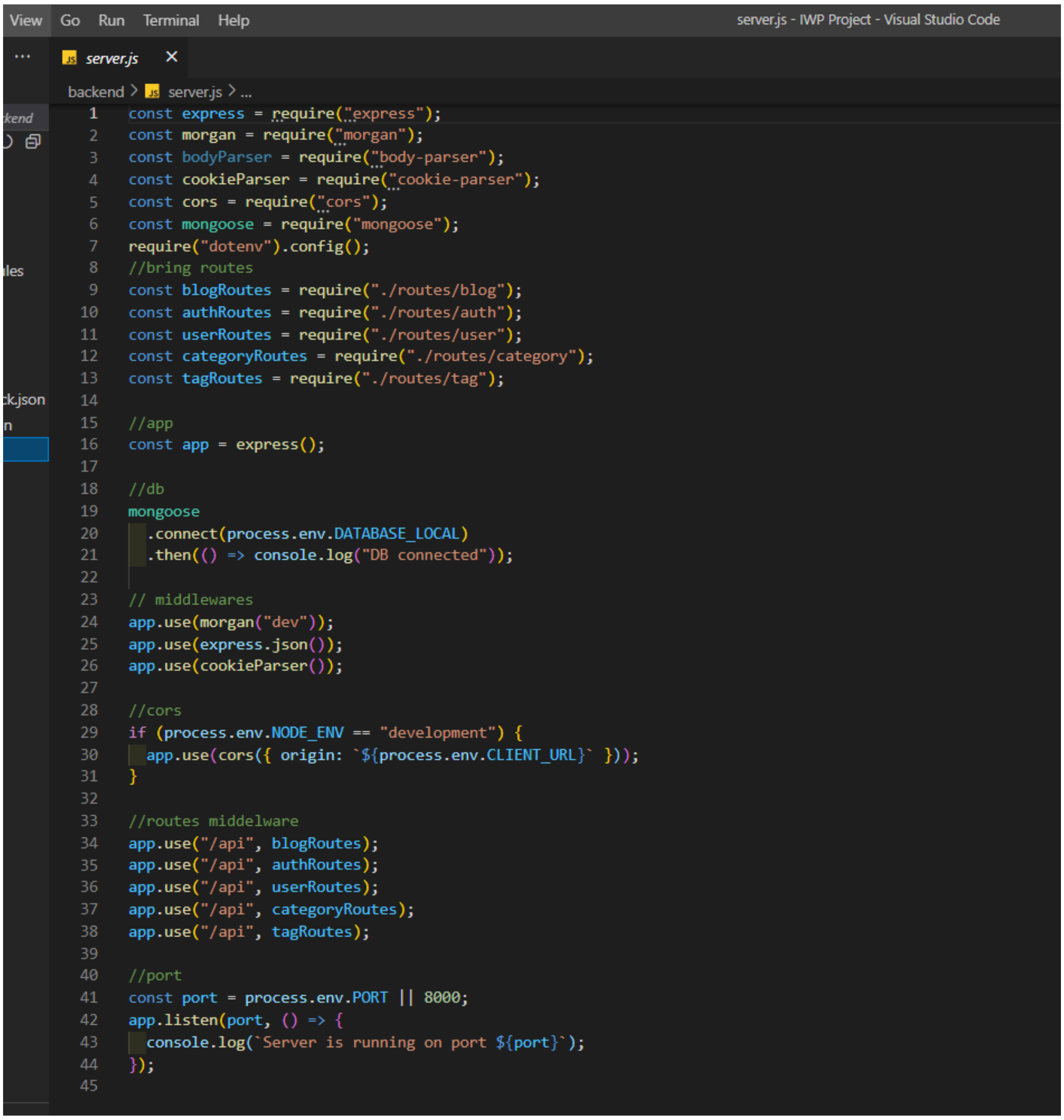
vii) Sign In and Sign Up pages

- The signin and the signup pages contain the general layout of the signin and signup components as defined earlier.

IMPORTANT CODE SCREENSHOTS

Backend

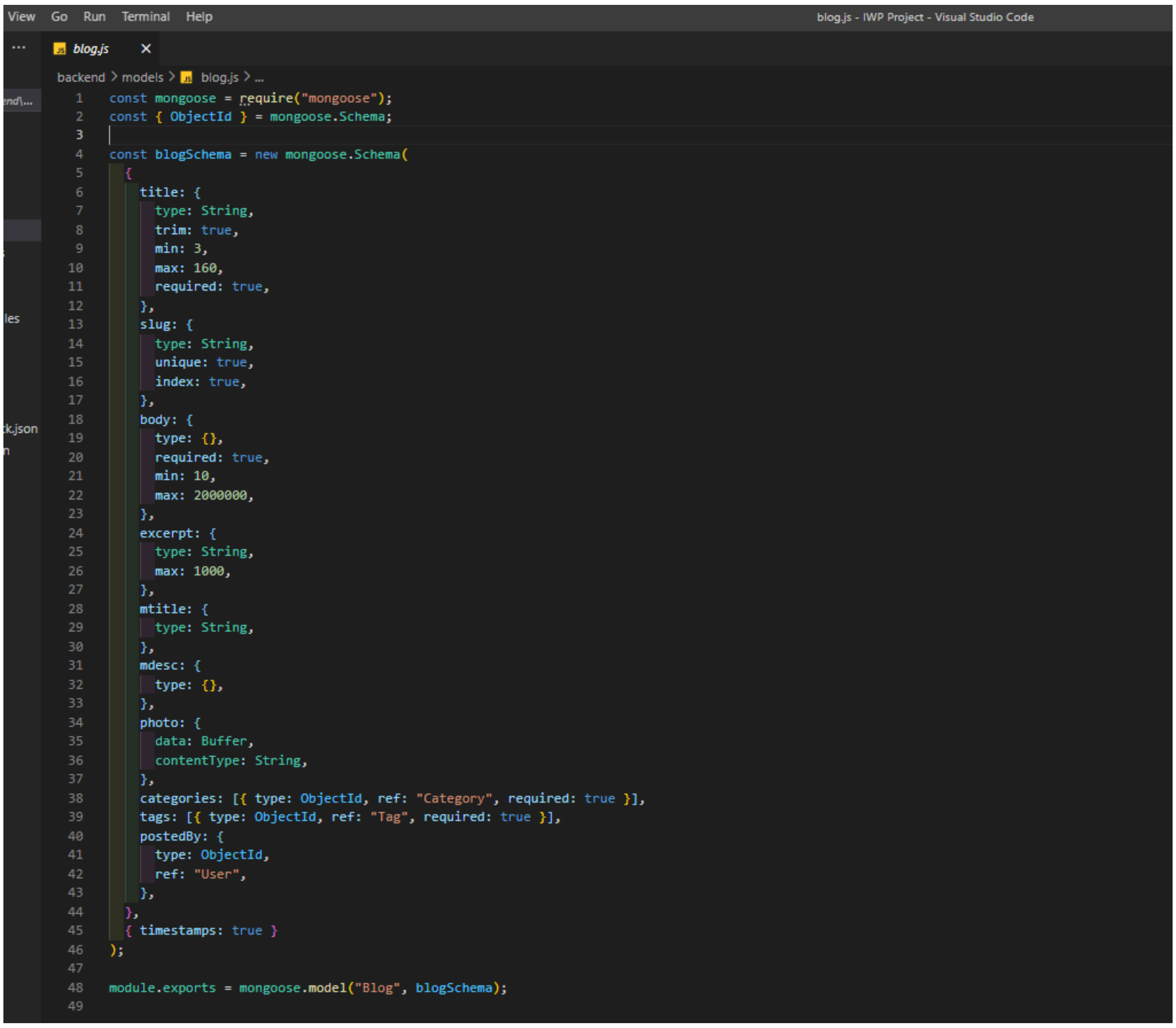
1. server.js



View Go Run Terminal Help
server.js - IWP Project - Visual Studio Code
server.js
backend > server.js > ...

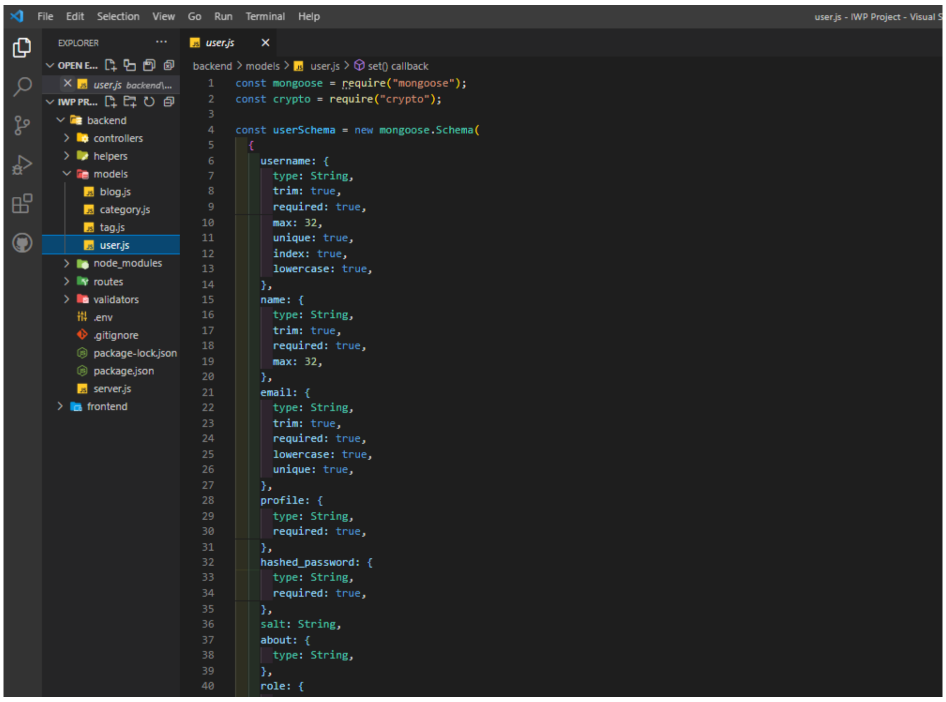
```javascript
const express = require("express");
const morgan = require("morgan");
const bodyParser = require("body-parser");
const cookieParser = require("cookie-parser");
const cors = require("cors");
const mongoose = require("mongoose");
require("dotenv").config();
//bring routes
const blogRoutes = require("./routes/blog");
const authRoutes = require("./routes/auth");
const userRoutes = require("./routes/user");
const categoryRoutes = require("./routes/category");
const tagRoutes = require("./routes/tag");

//app
const app = express();

//db
mongoose
  .connect(process.env.DATABASE_LOCAL)
  .then(() => console.log("DB connected"));

// middlewares
app.use(morgan("dev"));
app.use(express.json());
app.use(cookieParser());

//cors
if (process.env.NODE_ENV == "development") {
  app.use(cors({ origin: `${process.env.CLIENT_URL}` }));
}

//routes middelware
app.use("/api", blogRoutes);
app.use("/api", authRoutes);
app.use("/api", userRoutes);
app.use("/api", categoryRoutes);
app.use("/api", tagRoutes);

//port
const port = process.env.PORT || 8000;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

2. blog model

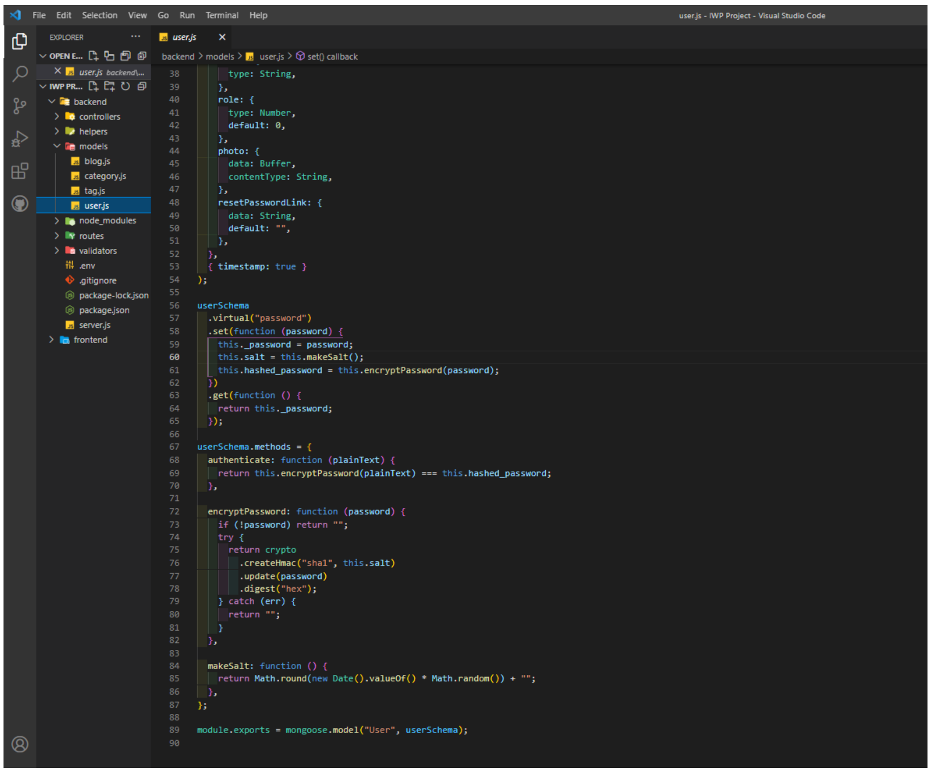

View Go Run Terminal Help
blog.js - IWP Project - Visual Studio Code
blog.js
backend > models > blog.js > ...

```
const mongoose = require("mongoose");
const { ObjectId } = mongoose.Schema;

const blogSchema = new mongoose.Schema(
  {
    title: {
      type: String,
      trim: true,
      min: 3,
      max: 160,
      required: true,
    },
    slug: {
      type: String,
      unique: true,
      index: true,
    },
    body: {
      type: {},
      required: true,
      min: 10,
      max: 2000000,
    },
    excerpt: {
      type: String,
      max: 1000,
    },
    mtitle: {
      type: String,
    },
    mdesc: {
      type: {},
    },
    photo: {
      data: Buffer,
      contentType: String,
    },
    categories: [{ type: ObjectId, ref: "Category", required: true }],
    tags: [{ type: ObjectId, ref: "Tag", required: true }],
    postedBy: {
      type: ObjectId,
      ref: "User",
    },
  },
  { timestamps: true }
);

module.exports = mongoose.model("Blog", blogSchema);
```

3. User Model

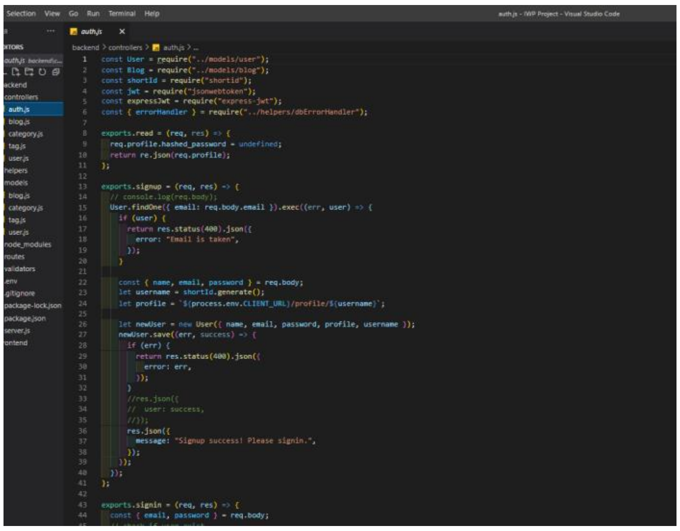


```
const mongoose = require("mongoose");
const crypto = require("crypto");

const userSchema = new mongoose.Schema(
  {
    username: {
      type: String,
      trim: true,
      required: true,
      max: 32,
      unique: true,
      index: true,
      lowercase: true,
    },
    name: {
      type: String,
      trim: true,
      required: true,
      max: 32,
    },
    email: {
      type: String,
      trim: true,
      required: true,
      lowercase: true,
      unique: true,
    },
    profile: {
      type: String,
      required: true,
    },
    hashed_password: {
      type: String,
      required: true,
    },
    salt: String,
    about: {
      type: String,
    },
    role: {
```

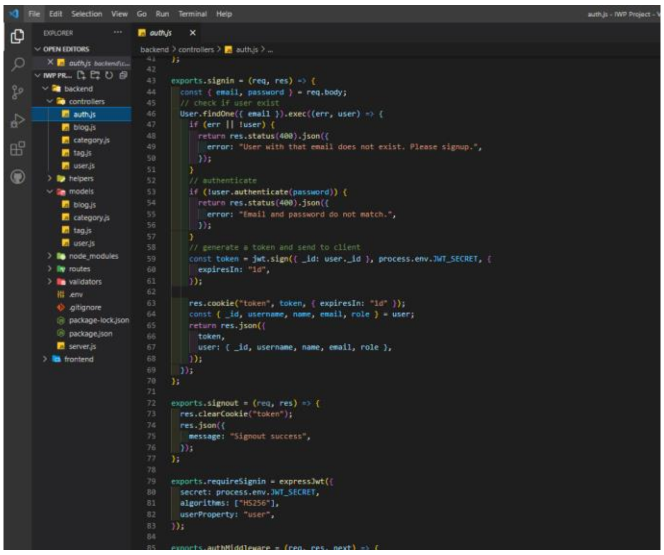


```
      type: String,
    },
    role: {
      type: Number,
      default: 0,
    },
    photo: {
      data: Buffer,
      contentType: String,
    },
    resetPasswordLink: {
      data: String,
      default: "",
    },
  },
  { timestamp: true }
);

userSchema
  .virtual("password")
  .set(function (password) {
    this._password = password;
    this.salt = this.makeSalt();
    this.hashed_password = this.encryptPassword(password);
  })
  .get(function () {
    return this._password;
  });

userSchema.methods = {
  authenticate: function (plainText) {
    return this.encryptPassword(plainText) === this.hashed_password;
  },

  encryptPassword: function (password) {
    if (!password) return "";
    try {
      return crypto
        .createHmac("sha1", this.salt)
        .update(password)
        .digest("hex");
    } catch (err) {
      return "";
    }
  },

  makeSalt: function () {
    return Math.round(new Date().valueOf() * Math.random()) + "";
  },
};

module.exports = mongoose.model("User", userSchema);
```

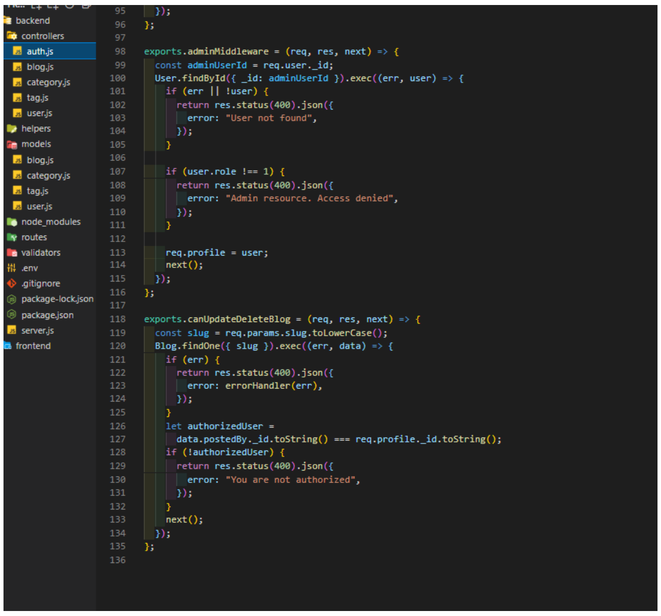
4. Auth Controller



```javascript
const User = require("../models/user");
const Blog = require("../models/blog");
const shortId = require("shortid");
const jwt = require("jsonwebtoken");
const expressJwt = require("express-jwt");
const { errorHandler } = require("../helpers/dbErrorHandler");

exports.read = (req, res) => {
  req.profile.hashed_password = undefined;
  return re.json(req.profile);
};

exports.signup = (req, res) => {
  // console.log(req.body);
  User.findOne({ email: req.body.email }).exec((err, user) => {
    if (user) {
      return res.status(400).json({
        error: "Email is taken",
      });
    }

    const { name, email, password } = req.body;
    let username = shortId.generate();
    let profile = `${process.env.CLIENT_URL}/profile/${username}`;

    let newUser = new User({ name, email, password, profile, username });
    newUser.save((err, success) => {
      if (err) {
        return res.status(400).json({
          error: err,
        });
      }
      //res.json({
      //  user: success,
      //});
      res.json({
        message: "Signup success! Please signin.",
      });
    });
  });
};

exports.signin = (req, res) => {
  const { email, password } = req.body;
```

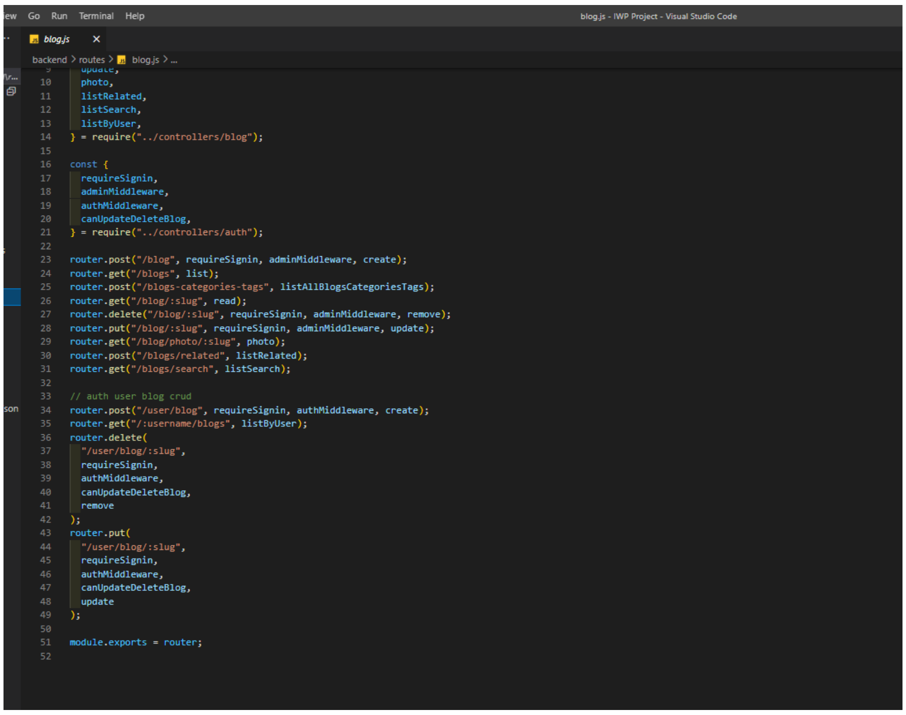


```javascript
};

exports.signin = (req, res) => {
  const { email, password } = req.body;
  // check if user exist
  User.findOne({ email }).exec((err, user) => {
    if (err || !user) {
      return res.status(400).json({
        error: "User with that email does not exist. Please signup.",
      });
    }
    // authenticate
    if (!user.authenticate(password)) {
      return res.status(400).json({
        error: "Email and password do not match.",
      });
    }
    // generate a token and send to client
    const token = jwt.sign({ _id: user._id }, process.env.JWT_SECRET, {
      expiresIn: "1d",
    });

    res.cookie("token", token, { expiresIn: "1d" });
    const { _id, username, name, email, role } = user;
    return res.json({
      token,
      user: { _id, username, name, email, role },
    });
  });
};

exports.signout = (req, res) => {
  res.clearCookie("token");
  res.json({
    message: "Signout success",
  });
};

exports.requireSignin = expressJwt({
  secret: process.env.JWT_SECRET,
  algorithms: ["HS256"],
  userProperty: "user",
});

exports.authMiddleware = (req, res, next) => {
```



```javascript
  });
};

exports.adminMiddleware = (req, res, next) => {
  const adminUserId = req.user._id;
  User.findById({ _id: adminUserId }).exec((err, user) => {
    if (err || !user) {
      return res.status(400).json({
        error: "User not found",
      });
    }

    if (user.role !== 1) {
      return res.status(400).json({
        error: "Admin resource. Access denied",
      });
    }

    req.profile = user;
    next();
  });
};

exports.canUpdateDeleteBlog = (req, res, next) => {
  const slug = req.params.slug.toLowerCase();
  Blog.findOne({ slug }).exec((err, data) => {
    if (err) {
      return res.status(400).json({
        error: errorHandler(err),
      });
    }
    let authorizedUser =
      data.postedBy._id.toString() === req.profile._id.toString();
    if (!authorizedUser) {
      return res.status(400).json({
        error: "You are not authorized",
      });
    }
    next();
  });
};
```

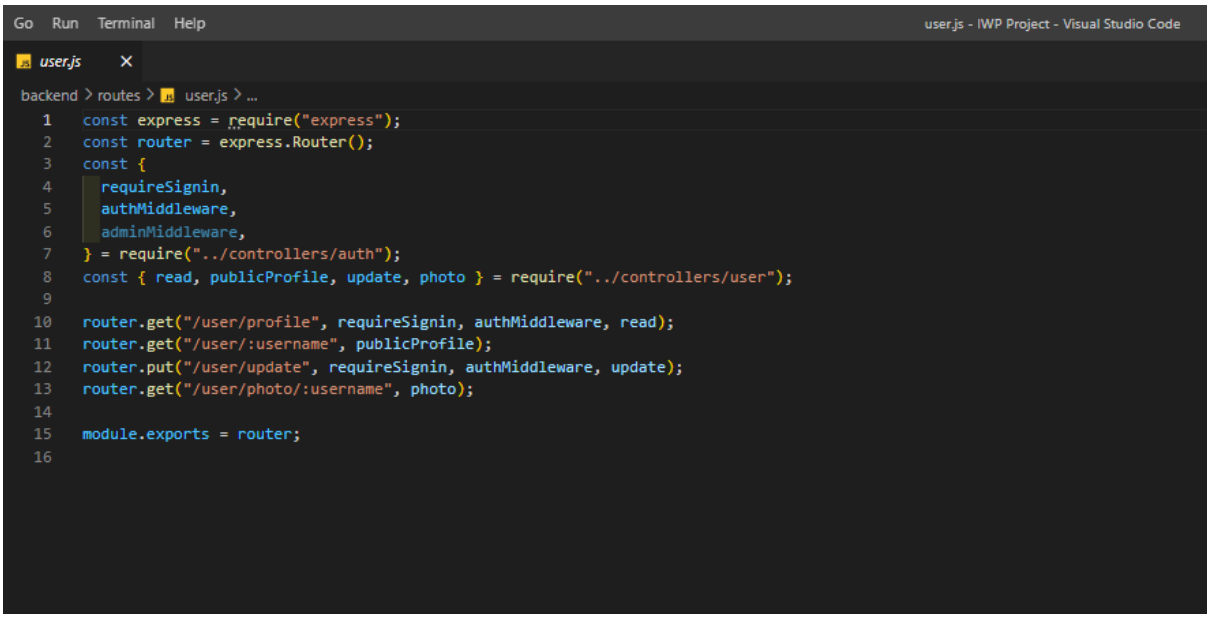
5. blog routes



iew Go Run Terminal Help
blog.js - IWP Project - Visual Studio Code
blog.js
backend > routes > blog.js > ...

```
  update,
  photo,
  listRelated,
  listSearch,
  listByUser,
} = require("../controllers/blog");

const {
  requireSignin,
  adminMiddleware,
  authMiddleware,
  canUpdateDeleteBlog,
} = require("../controllers/auth");

router.post("/blog", requireSignin, adminMiddleware, create);
router.get("/blogs", list);
router.post("/blogs-categories-tags", listAllBlogsCategoriesTags);
router.get("/blog/:slug", read);
router.delete("/blog/:slug", requireSignin, adminMiddleware, remove);
router.put("/blog/:slug", requireSignin, adminMiddleware, update);
router.get("/blog/photo/:slug", photo);
router.post("/blogs/related", listRelated);
router.get("/blogs/search", listSearch);

// auth user blog crud
router.post("/user/blog", requireSignin, authMiddleware, create);
router.get("/:username/blogs", listByUser);
router.delete(
  "/user/blog/:slug",
  requireSignin,
  authMiddleware,
  canUpdateDeleteBlog,
  remove
);
router.put(
  "/user/blog/:slug",
  requireSignin,
  authMiddleware,
  canUpdateDeleteBlog,
  update
);

module.exports = router;
```

6. User Routes

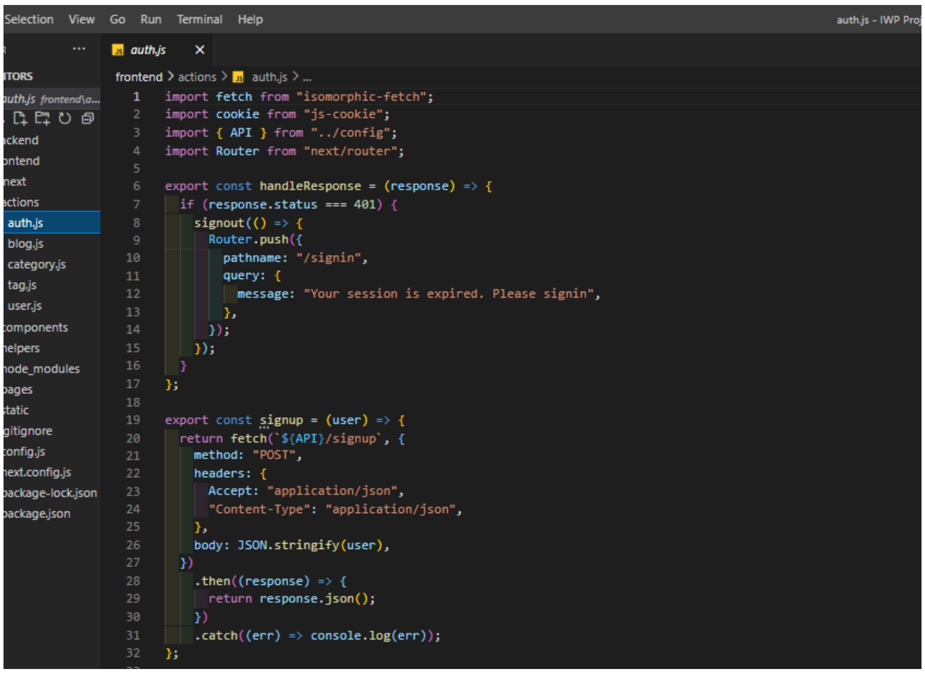


Go Run Terminal Help
user.js - IWP Project - Visual Studio Code
user.js
backend > routes > user.js > ...

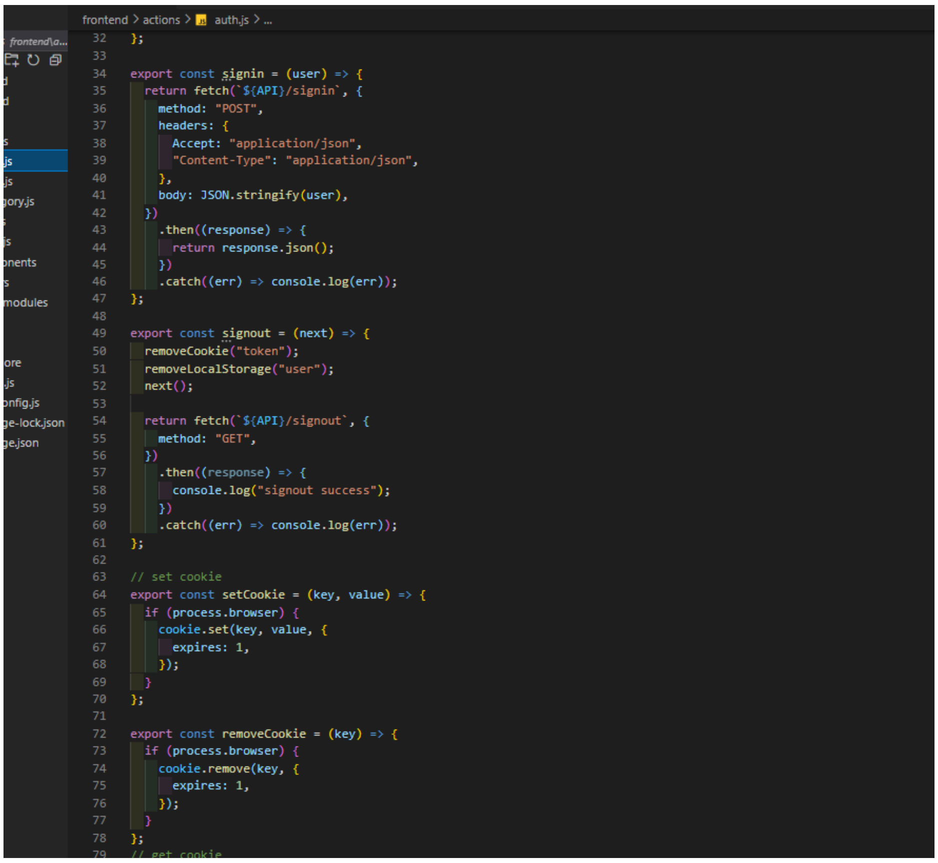
```
const express = require("express");
const router = express.Router();
const {
  requireSignin,
  authMiddleware,
  adminMiddleware,
} = require("../controllers/auth");
const { read, publicProfile, update, photo } = require("../controllers/user");

router.get("/user/profile", requireSignin, authMiddleware, read);
router.get("/user/:username", publicProfile);
router.put("/user/update", requireSignin, authMiddleware, update);
router.get("/user/photo/:username", photo);

module.exports = router;
```

Frontend

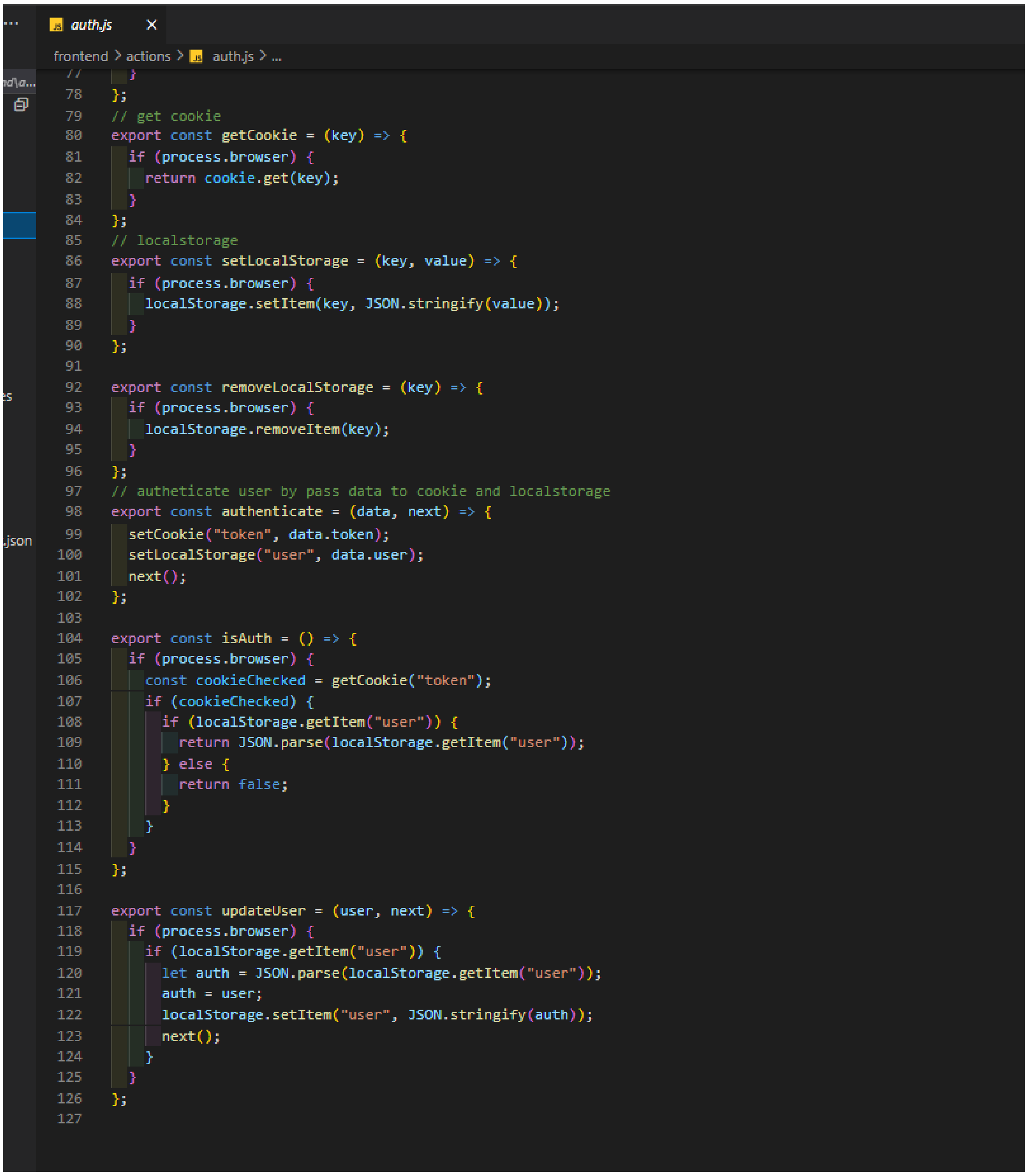
1.



```javascript
import fetch from "isomorphic-fetch";
import cookie from "js-cookie";
import { API } from "../config";
import Router from "next/router";

export const handleResponse = (response) => {
  if (response.status === 401) {
    signout(() => {
      Router.push({
        pathname: "/signin",
        query: {
          message: "Your session is expired. Please signin",
        },
      });
    });
  }
};

export const signup = (user) => {
  return fetch(`${API}/signup`, {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(user),
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};
```



```javascript
};

export const signin = (user) => {
  return fetch(`${API}/signin`, {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(user),
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};

export const signout = (next) => {
  removeCookie("token");
  removeLocalStorage("user");
  next();

  return fetch(`${API}/signout`, {
    method: "GET",
  })
    .then((response) => {
      console.log("signout success");
    })
    .catch((err) => console.log(err));
};

// set cookie
export const setCookie = (key, value) => {
  if (process.browser) {
    cookie.set(key, value, {
      expires: 1,
    });
  }
};

export const removeCookie = (key) => {
  if (process.browser) {
    cookie.remove(key, {
      expires: 1,
    });
  }
};
// get cookie
```

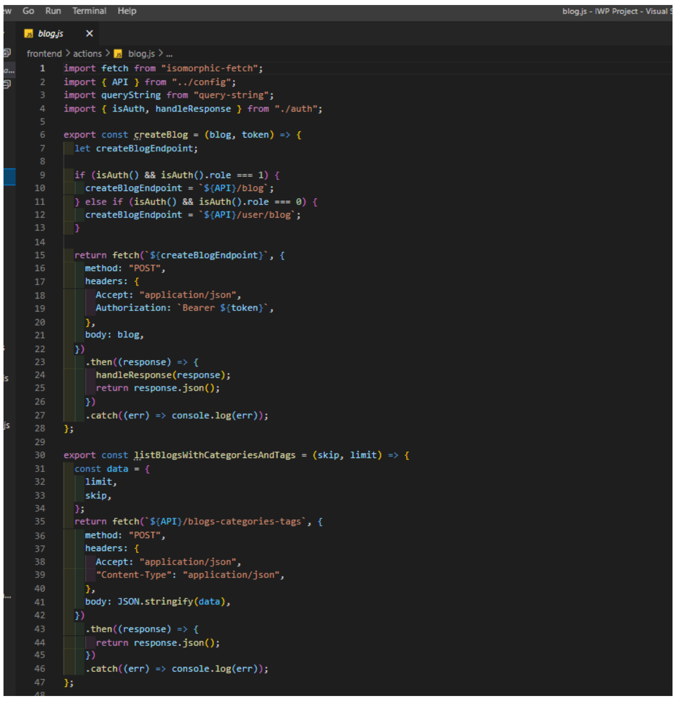


auth.js

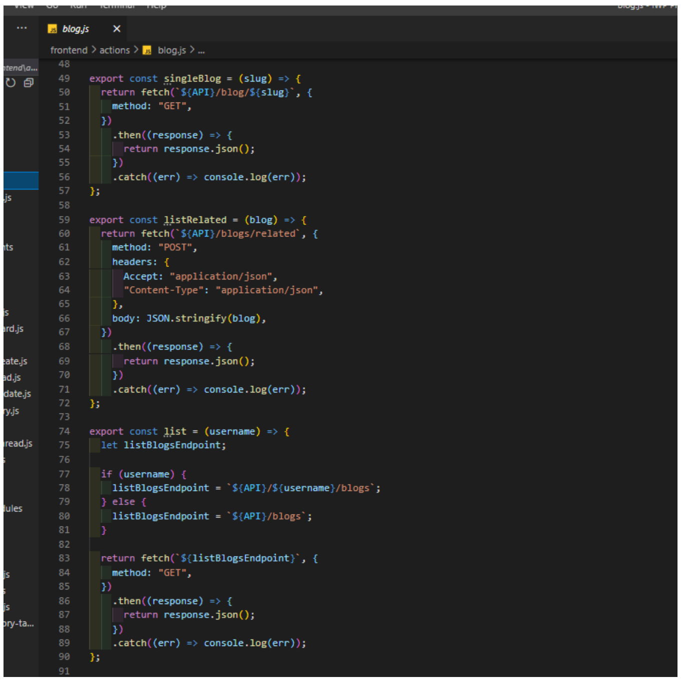
frontend > actions > auth.js > ...

```
  }
};
// get cookie
export const getCookie = (key) => {
  if (process.browser) {
    return cookie.get(key);
  }
};
// localstorage
export const setLocalStorage = (key, value) => {
  if (process.browser) {
    localStorage.setItem(key, JSON.stringify(value));
  }
};

export const removeLocalStorage = (key) => {
  if (process.browser) {
    localStorage.removeItem(key);
  }
};
// autheticate user by pass data to cookie and localstorage
export const authenticate = (data, next) => {
  setCookie("token", data.token);
  setLocalStorage("user", data.user);
  next();
};

export const isAuth = () => {
  if (process.browser) {
    const cookieChecked = getCookie("token");
    if (cookieChecked) {
      if (localStorage.getItem("user")) {
        return JSON.parse(localStorage.getItem("user"));
      } else {
        return false;
      }
    }
  }
};

export const updateUser = (user, next) => {
  if (process.browser) {
    if (localStorage.getItem("user")) {
      let auth = JSON.parse(localStorage.getItem("user"));
      auth = user;
      localStorage.setItem("user", JSON.stringify(auth));
      next();
    }
  }
};
```

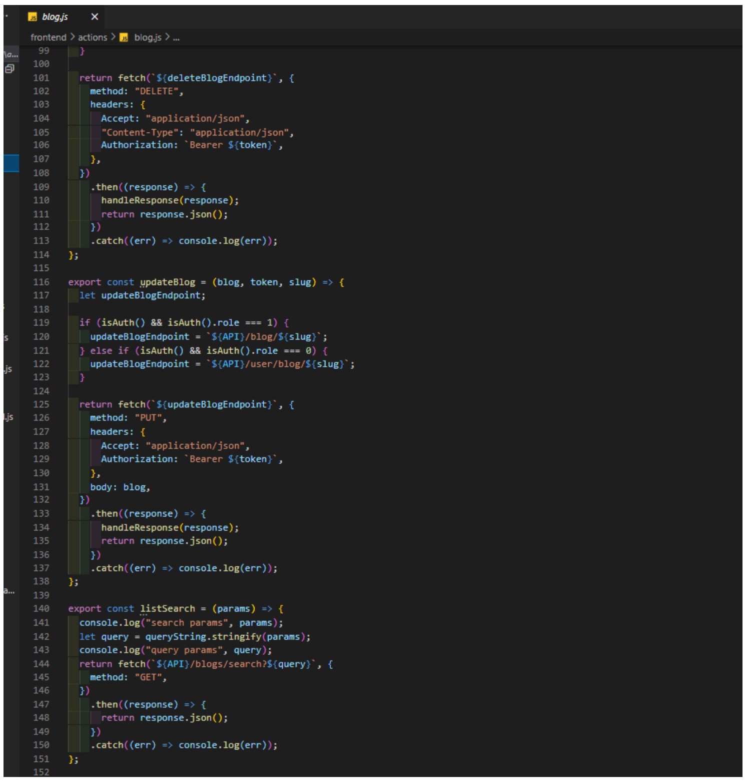
2. Blog Actions



```
import fetch from "isomorphic-fetch";
import { API } from "../config";
import queryString from "query-string";
import { isAuth, handleResponse } from "./auth";

export const createBlog = (blog, token) => {
  let createBlogEndpoint;

  if (isAuth() && isAuth().role === 1) {
    createBlogEndpoint = `${API}/blog`;
  } else if (isAuth() && isAuth().role === 0) {
    createBlogEndpoint = `${API}/user/blog`;
  }

  return fetch(`${createBlogEndpoint}`, {
    method: "POST",
    headers: {
      Accept: "application/json",
      Authorization: `Bearer ${token}`,
    },
    body: blog,
  })
    .then((response) => {
      handleResponse(response);
      return response.json();
    })
    .catch((err) => console.log(err));
};

export const listBlogsWithCategoriesAndTags = (skip, limit) => {
  const data = {
    limit,
    skip,
  };
  return fetch(`${API}/blogs-categories-tags`, {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(data),
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};
```



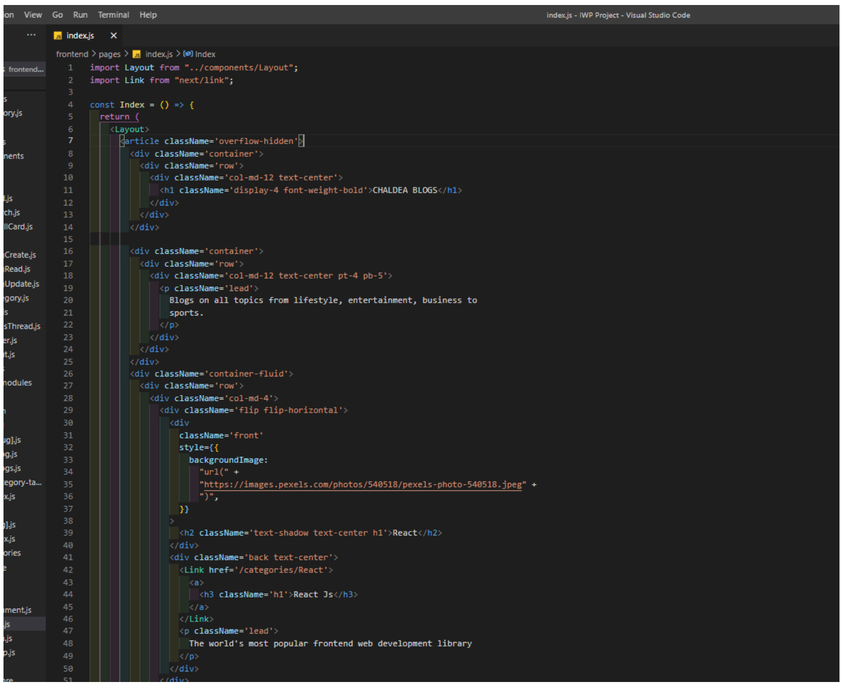
```

export const singleBlog = (slug) => {
  return fetch(`${API}/blog/${slug}`, {
    method: "GET",
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};

export const listRelated = (blog) => {
  return fetch(`${API}/blogs/related`, {
    method: "POST",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
    },
    body: JSON.stringify(blog),
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};

export const list = (username) => {
  let listBlogsEndpoint;

  if (username) {
    listBlogsEndpoint = `${API}/${username}/blogs`;
  } else {
    listBlogsEndpoint = `${API}/blogs`;
  }

  return fetch(`${listBlogsEndpoint}`, {
    method: "GET",
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};
```


blog.js

frontend > actions > blog.js > ...

```javascript
  }

  return fetch(`${deleteBlogEndpoint}`, {
    method: "DELETE",
    headers: {
      Accept: "application/json",
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    },
  })
    .then((response) => {
      handleResponse(response);
      return response.json();
    })
    .catch((err) => console.log(err));
};

export const updateBlog = (blog, token, slug) => {
  let updateBlogEndpoint;

  if (isAuth() && isAuth().role === 1) {
    updateBlogEndpoint = `${API}/blog/${slug}`;
  } else if (isAuth() && isAuth().role === 0) {
    updateBlogEndpoint = `${API}/user/blog/${slug}`;
  }

  return fetch(`${updateBlogEndpoint}`, {
    method: "PUT",
    headers: {
      Accept: "application/json",
      Authorization: `Bearer ${token}`,
    },
    body: blog,
  })
    .then((response) => {
      handleResponse(response);
      return response.json();
    })
    .catch((err) => console.log(err));
};

export const listSearch = (params) => {
  console.log("search params", params);
  let query = queryString.stringify(params);
  console.log("query params", query);
  return fetch(`${API}/blogs/search?${query}`, {
    method: "GET",
  })
    .then((response) => {
      return response.json();
    })
    .catch((err) => console.log(err));
};

```

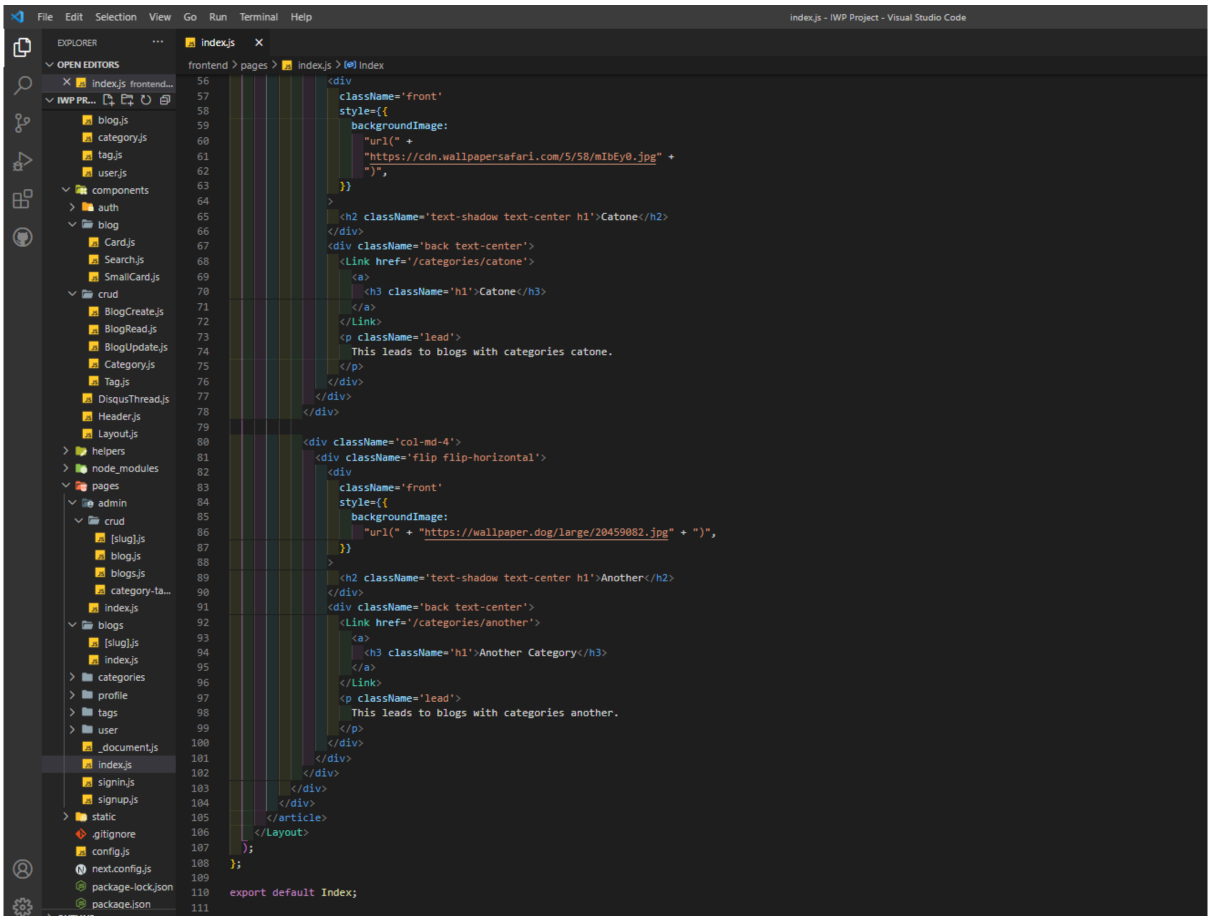
3. Landing Page


index.js - IWP Project - Visual Studio Code

index.js

frontend > pages > index.js > Index

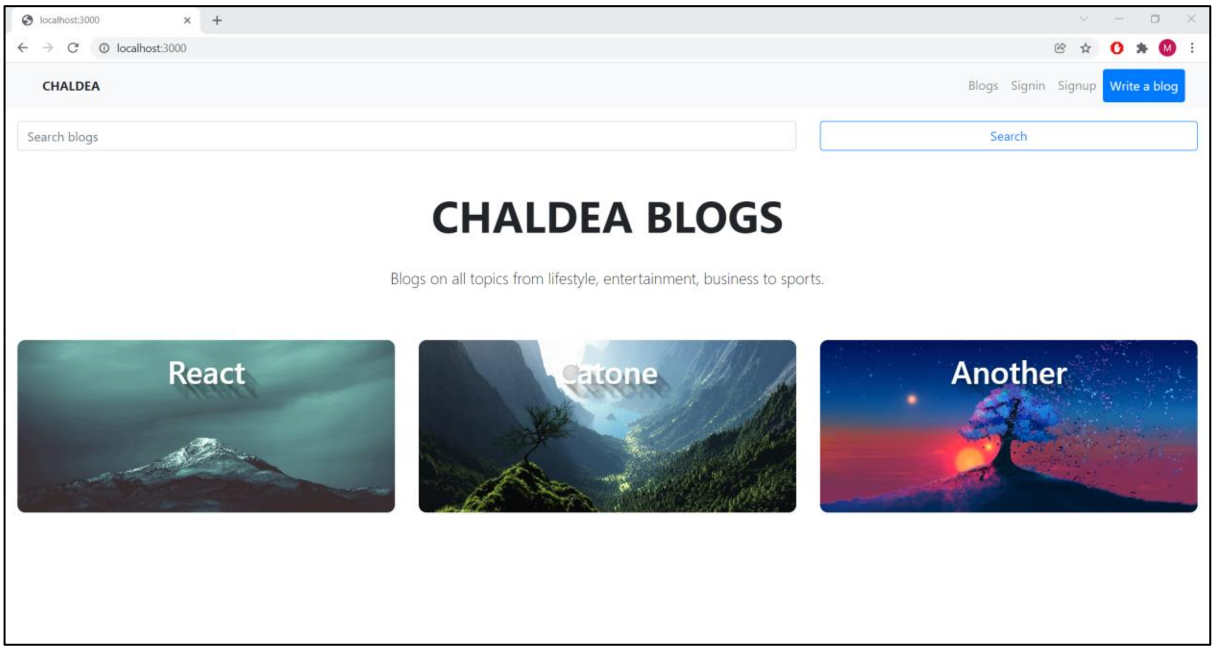
```javascript
import Layout from "../components/Layout";
import Link from "next/link";

const Index = () => {
  return (
    <Layout>
      <article className='overflow-hidden'>
        <div className='container'>
          <div className='row'>
            <div className='col-md-12 text-center'>
              <h1 className='display-4 font-weight-bold'>CHALDEA BLOGS</h1>
            </div>
          </div>
        </div>

        <div className='container'>
          <div className='row'>
            <div className='col-md-12 text-center pt-4 pb-5'>
              <p className='lead'>
                Blogs on all topics from lifestyle, entertainment, business to
                sports.
              </p>
            </div>
          </div>
        </div>
        <div className='container-fluid'>
          <div className='row'>
            <div className='col-md-4'>
              <div className='flip flip-horizontal'>
                <div
                  className='front'
                  style={{
                    backgroundImage:
                      "url(" +
                      "https://images.pexels.com/photos/540518/pexels-photo-540518.jpeg" +
                      ")",
                  }}
                >
                  <h2 className='text-shadow text-center h1'>React</h2>
                </div>
                <div className='back text-center'>
                  <Link href='/categories/React'>
                    <a>
                      <h3 className='h1'>React Js</h3>
                    </a>
                  </Link>
                  <p className='lead'>
                    The world's most popular frontend web development library
                  </p>
                </div>
              </div>
```


```
                <div
                  className='front'
                  style={{
                    backgroundImage:
                      "url(" +
                      "https://cdn.wallpapersafari.com/5/58/mIbEy0.jpg" +
                      ")",
                  }}
                >
                  <h2 className='text-shadow text-center h1'>Catone</h2>
                </div>
                <div className='back text-center'>
                  <Link href='/categories/catone'>
                    <a>
                      <h3 className='h1'>Catone</h3>
                    </a>
                  </Link>
                  <p className='lead'>
                    This leads to blogs with categories catone.
                  </p>
                </div>
              </div>
            </div>

            <div className='col-md-4'>
              <div className='flip flip-horizontal'>
                <div
                  className='front'
                  style={{
                    backgroundImage:
                      "url(" + "https://wallpaper.dog/large/20459082.jpg" + ")",
                  }}
                >
                  <h2 className='text-shadow text-center h1'>Another</h2>
                </div>
                <div className='back text-center'>
                  <Link href='/categories/another'>
                    <a>
                      <h3 className='h1'>Another Category</h3>
                    </a>
                  </Link>
                  <p className='lead'>
                    This leads to blogs with categories another.
                  </p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </article>
    </Layout>
  );
};

export default Index;
```

Many other pages are present in the project. The images given above are some of them.

RESULTS

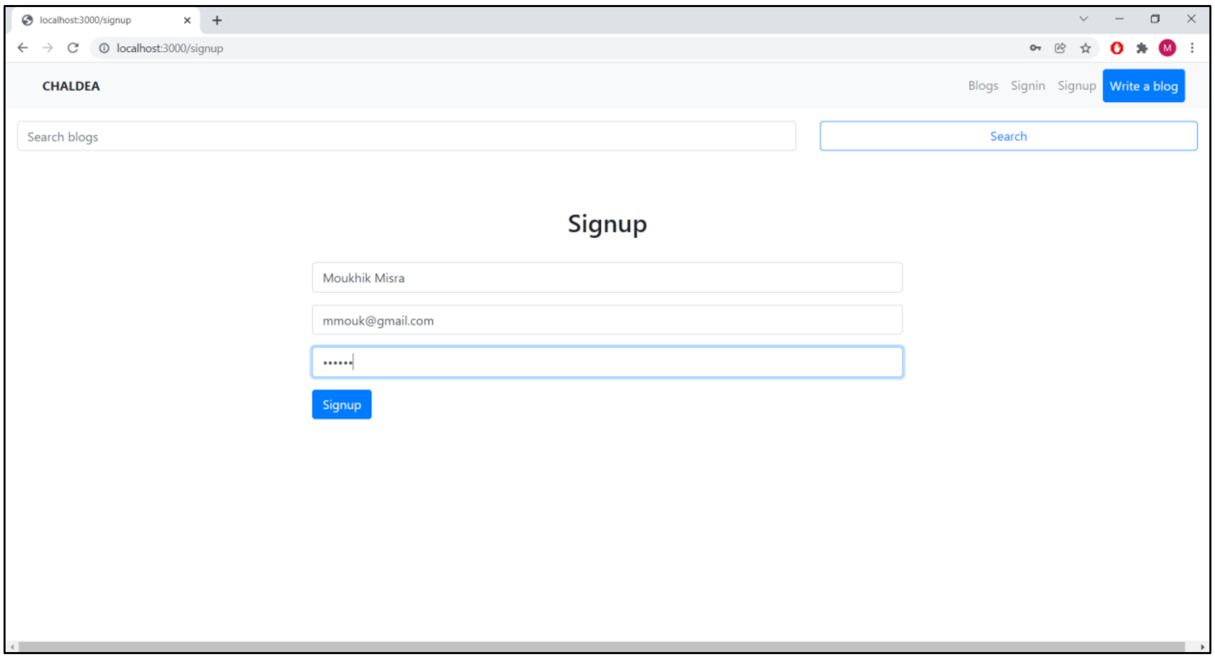
1 Landing Page



CHALDEA BLOGS

Blogs on all topics from lifestyle, entertainment, business to sports.

2. Signup Page

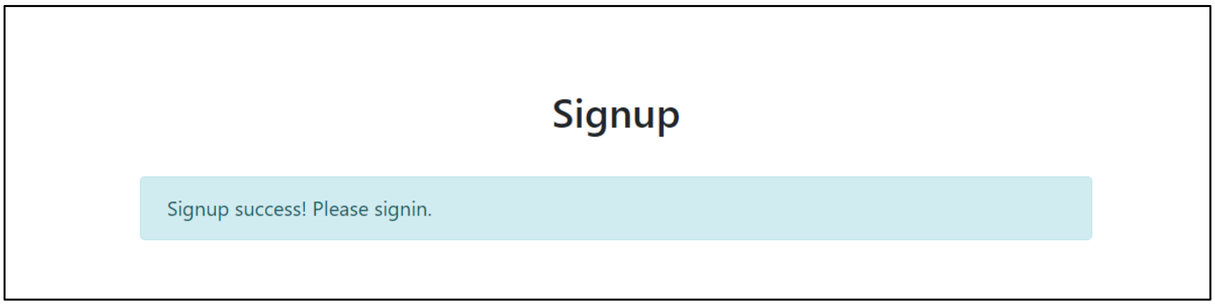


Moukhik Misra

mmouk@gmail.com

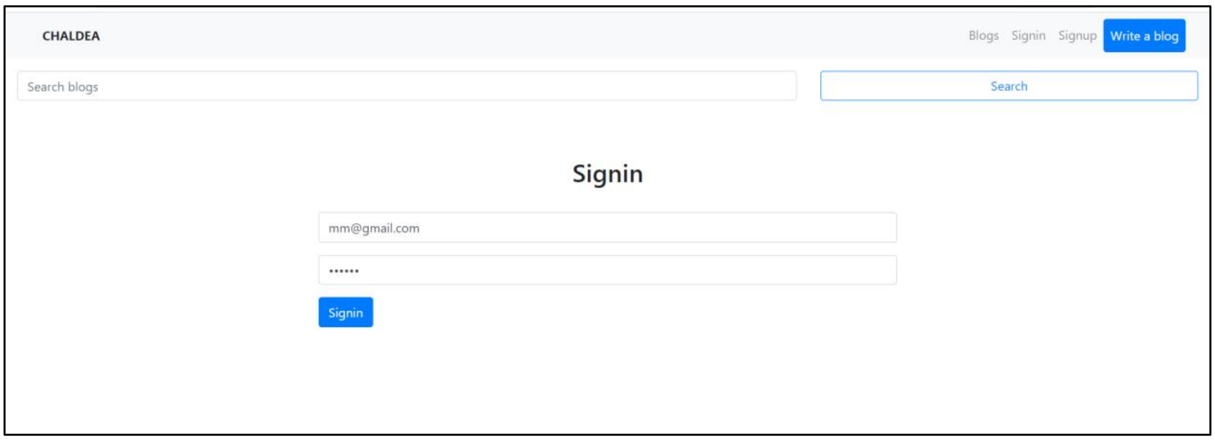
••••••

Successful Signup


Signup

Signup success! Please signin.

3. Admin SignIn

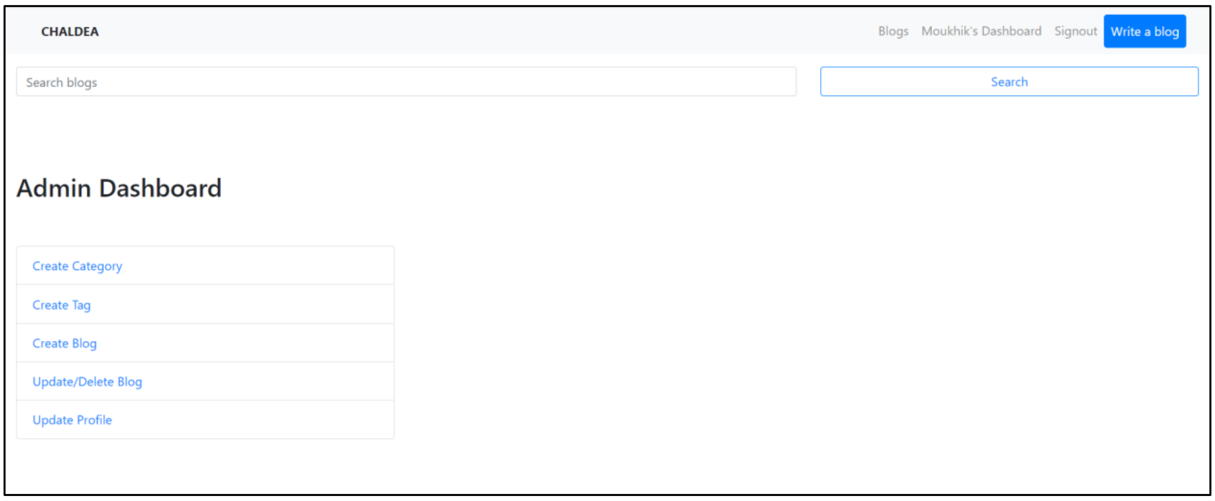


Signin

mm@gmail.com

••••••

4. Admin Dashboard

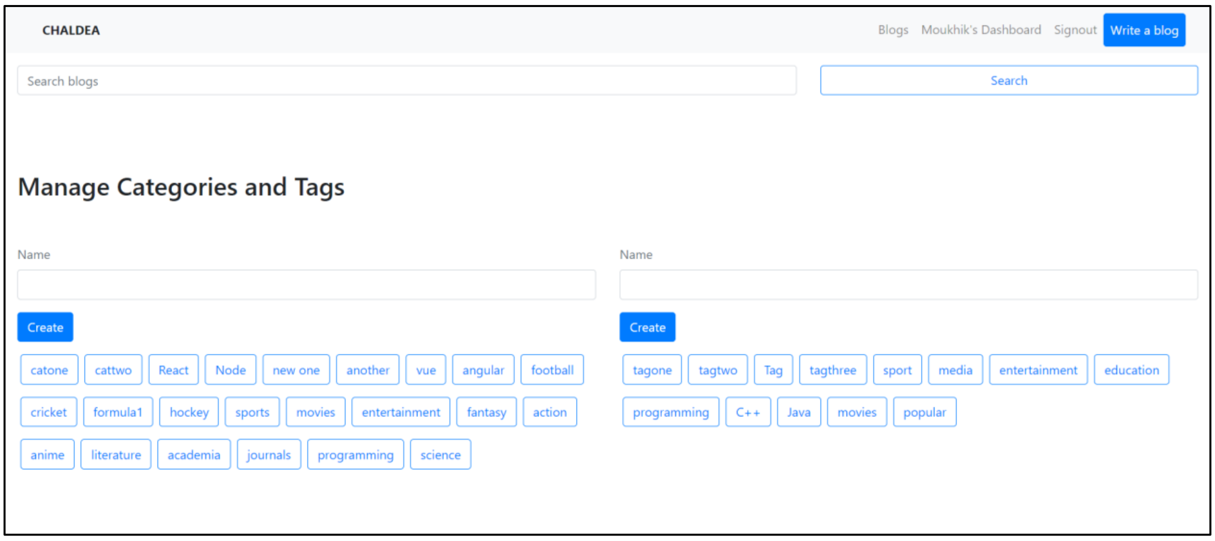


Admin Dashboard

- Create Category
- Create Tag
- Create Blog
- Update/Delete Blog
- Update Profile

5. Create Category and Tags

Original:


CHALDEA

Blogs Moukhik's Dashboard Signout Write a blog

Search blogs | Search

Manage Categories and Tags

Name

Create

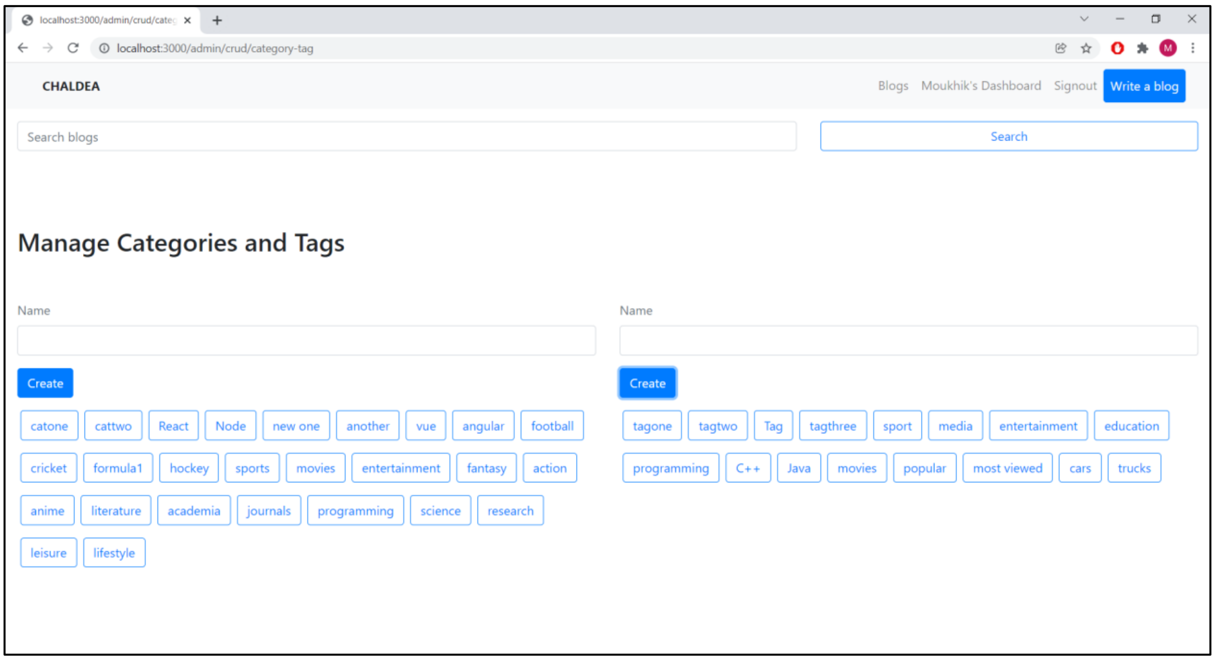
catone cattwo React Node new one another vue angular football cricket formula1 hockey sports movies entertainment fantasy action anime literature academia journals programming science

Name

Create

tagone tagtwo Tag tagthree sport media entertainment education programming C++ Java movies popular

New:


localhost:3000/admin/crud/category-tag

CHALDEA

Blogs Moukhik's Dashboard Signout Write a blog

Search blogs | Search

Manage Categories and Tags

Name

Create

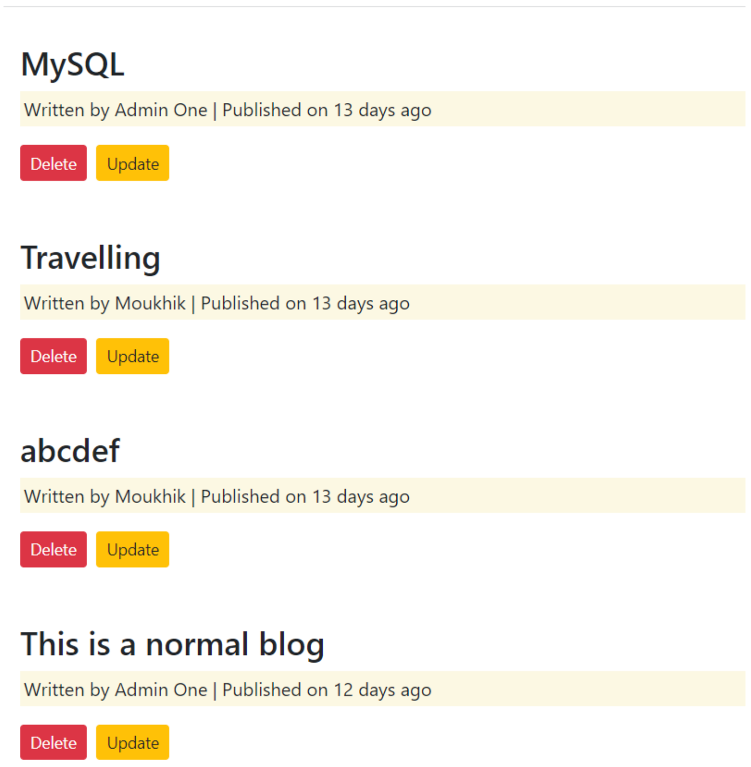
catone cattwo React Node new one another vue angular football cricket formula1 hockey sports movies entertainment fantasy action anime literature academia journals programming science research leisure lifestyle

Name

Create

tagone tagtwo Tag tagthree sport media entertainment education programming C++ Java movies popular most viewed cars trucks

Delete Blog (Update and Create Blog results will come in the user section)


MySQL

Written by Admin One | Published on 13 days ago

Travelling

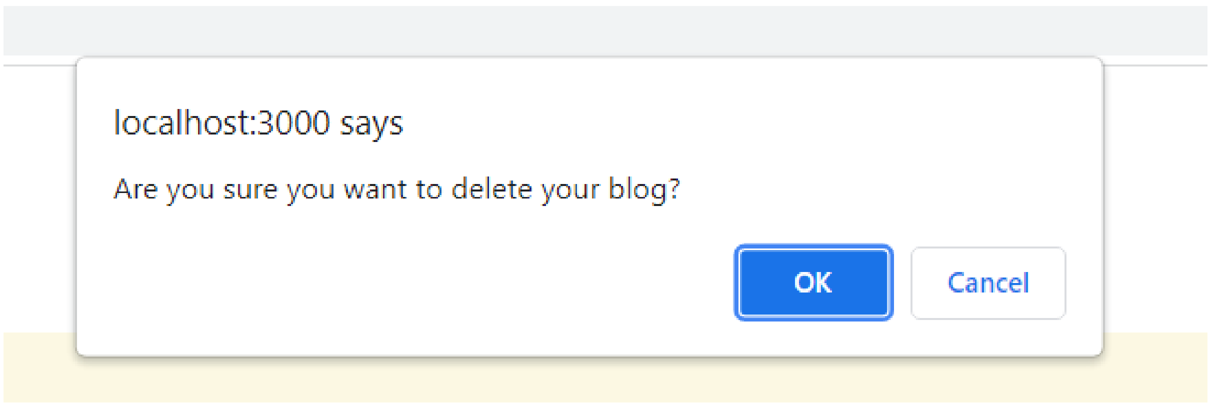
Written by Moukhik | Published on 13 days ago

abcdef

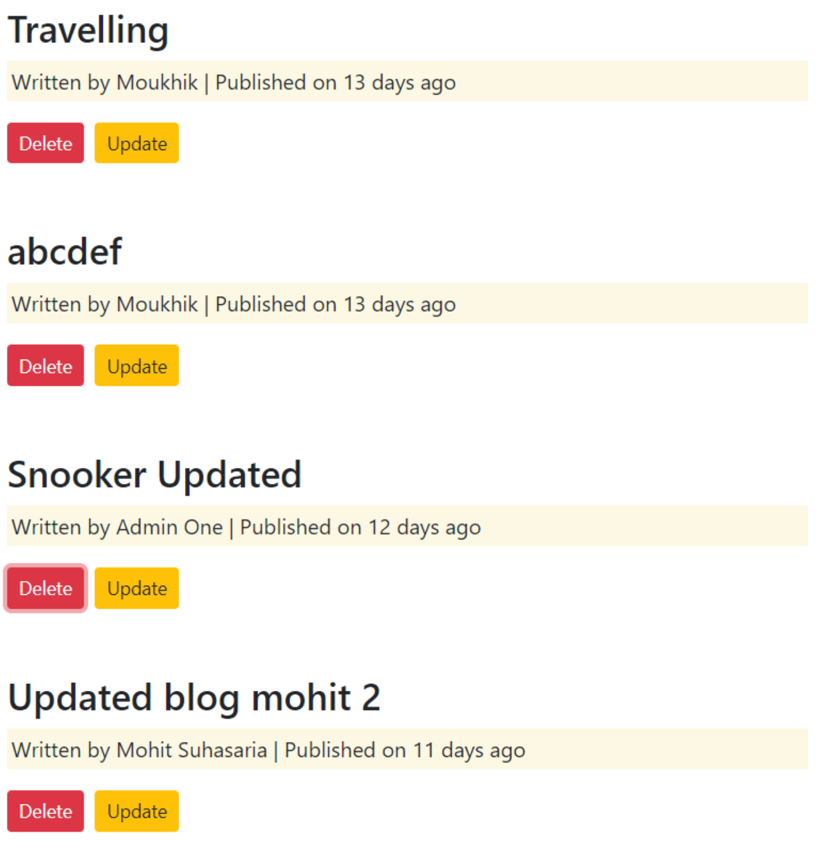
Written by Moukhik | Published on 13 days ago

This is a normal blog

Written by Admin One | Published on 12 days ago


localhost:3000 says

Are you sure you want to delete your blog?


Travelling

Written by Moukhik | Published on 13 days ago

abcdef

Written by Moukhik | Published on 13 days ago

Snooker Updated

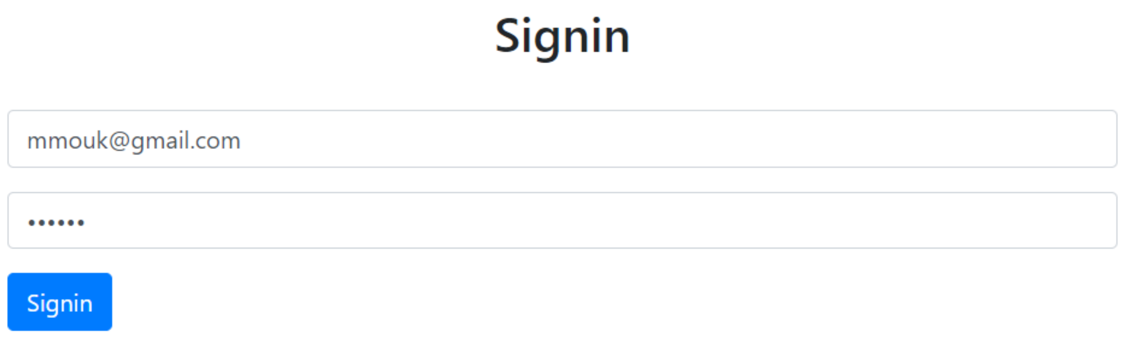
Written by Admin One | Published on 12 days ago

Updated blog mohit 2

Written by Mohit Suhasaria | Published on 11 days ago

This is a normal blog by Admin One has been Deleted

User Sign In:

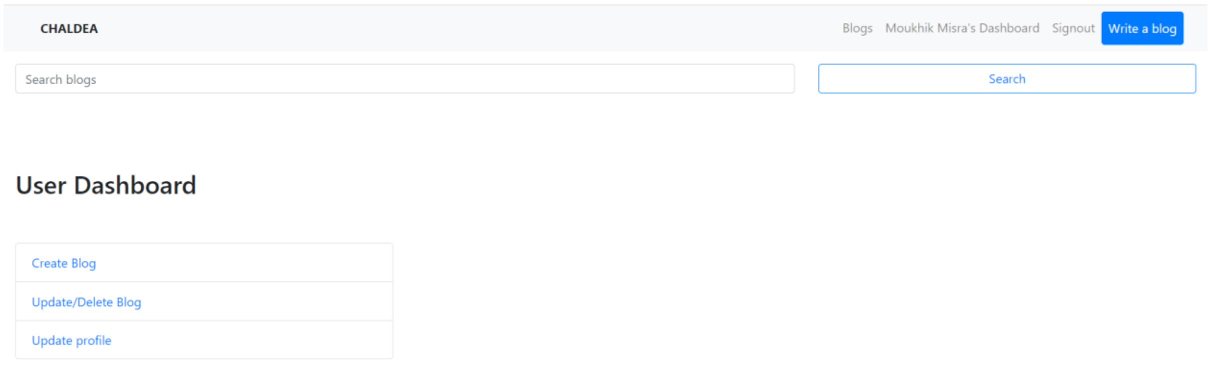

Signin

mmouk@gmail.com

••••••

New created user Moukhik Misra (not admin)

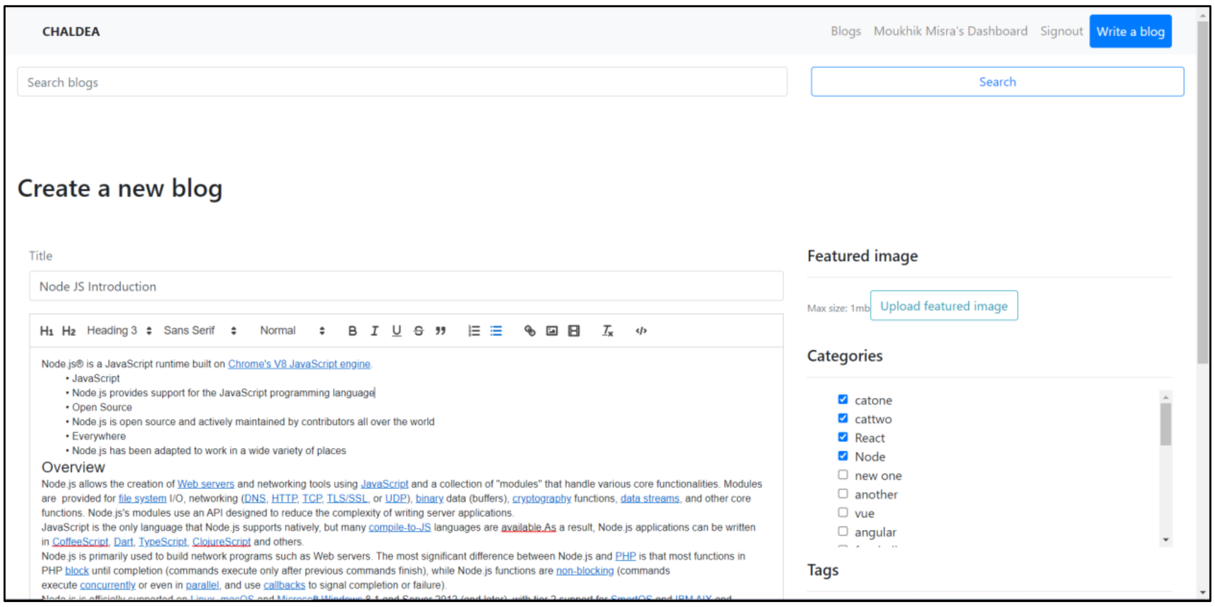
User Dashboard:



User Dashboard

- Create Blog
- Update/Delete Blog
- Update profile

Create Blog (Content, Categories, Tags, Image)



Create a new blog

Title: Node JS Introduction

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

- JavaScript
- Node.js provides support for the JavaScript programming language
- Open Source
- Node.js is open source and actively maintained by contributors all over the world
- Everywhere
- Node.js has been adapted to work in a wide variety of places

Overview

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.
JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available.As a result, Node.js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.
Node.js is primarily used to build network programs such as Web servers. The most significant difference between Node.js and PHP is that most functions in PHP block until completion (commands execute only after previous commands finish), while Node.js functions are non-blocking (commands execute concurrently or even in parallel, and use callbacks to signal completion or failure).

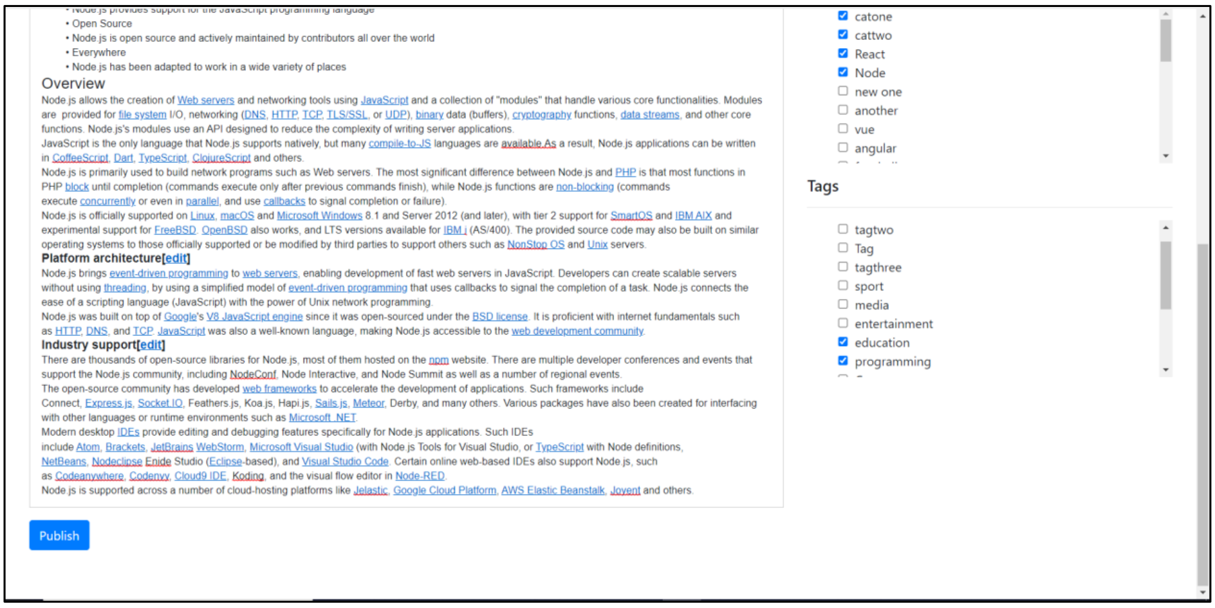
Featured image

Max size: 1mb

Categories

- [x] catone
- [x] cattwo
- [x] React
- [x] Node
- [ ] new one
- [ ] another
- [ ] vue
- [ ] angular

Tags


- Node.js provides support for the JavaScript programming language
- Open Source
- Node.js is open source and actively maintained by contributors all over the world
- Everywhere
- Node.js has been adapted to work in a wide variety of places

Overview

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.
JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available.As a result, Node.js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.
Node.js is primarily used to build network programs such as Web servers. The most significant difference between Node.js and PHP is that most functions in PHP block until completion (commands execute only after previous commands finish), while Node.js functions are non-blocking (commands execute concurrently or even in parallel, and use callbacks to signal completion or failure).
Node.js is officially supported on Linux, macOS and Microsoft Windows 8.1 and Server 2012 (and later), with tier 2 support for SmartOS and IBM AIX and experimental support for FreeBSD. OpenBSD also works, and LTS versions available for IBM i (AS/400). The provided source code may also be built on similar operating systems to those officially supported or be modified by third parties to support others such as NonStop OS and Unix servers.

Platform architecture[edit]

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js connects the ease of a scripting language (JavaScript) with the power of Unix network programming.
Node.js was built on top of Google's V8 JavaScript engine since it was open-sourced under the BSD license. It is proficient with internet fundamentals such as HTTP, DNS, and TCP. JavaScript was also a well-known language, making Node.js accessible to the web development community.

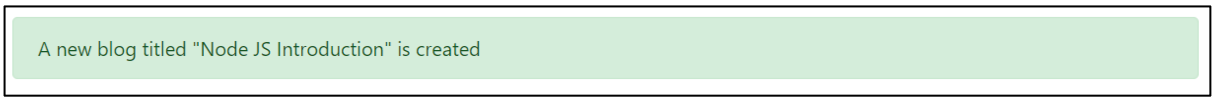
Industry support[edit]

There are thousands of open-source libraries for Node.js, most of them hosted on the npm website. There are multiple developer conferences and events that support the Node.js community, including NodeConf, Node Interactive, and Node Summit as well as a number of regional events.
The open-source community has developed web frameworks to accelerate the development of applications. Such frameworks include Connect, Express.js, Socket.IO, Feathers.js, Koa.js, Hapi.js, Sails.js, Meteor, Derby, and many others. Various packages have also been created for interfacing with other languages or runtime environments such as Microsoft .NET.
Modern desktop IDEs provide editing and debugging features specifically for Node.js applications. Such IDEs include Atom, Brackets, JetBrains WebStorm, Microsoft Visual Studio (with Node.js Tools for Visual Studio, or TypeScript with Node definitions, NetBeans, Nodeclipse Enide Studio (Eclipse-based), and Visual Studio Code. Certain online web-based IDEs also support Node.js, such as Codeanywhere, Codenvy, Cloud9 IDE, Koding, and the visual flow editor in Node-RED.
Node.js is supported across a number of cloud-hosting platforms like Jelastic, Google Cloud Platform, AWS Elastic Beanstalk, Joyent and others.

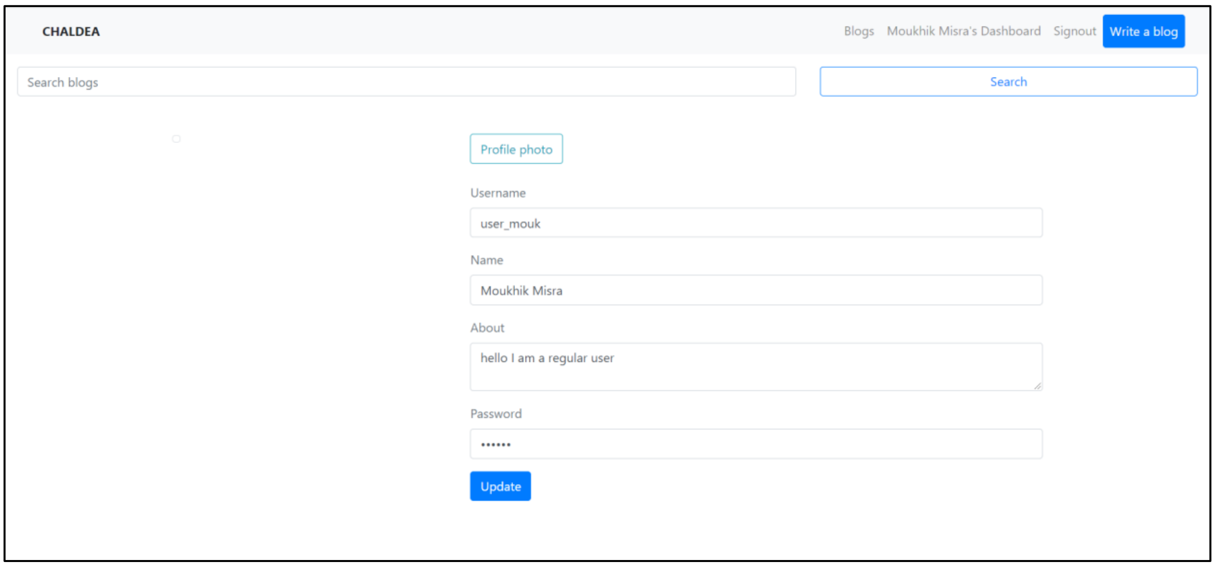
- [✓] catone
- [✓] cattwo
- [✓] React
- [✓] Node
- [ ] new one
- [ ] another
- [ ] vue
- [ ] angular

Tags

- [ ] tagtwo
- [ ] Tag
- [ ] tagthree
- [ ] sport
- [ ] media
- [ ] entertainment
- [✓] education
- [✓] programming


A new blog titled "Node JS Introduction" is created

Update User profile:



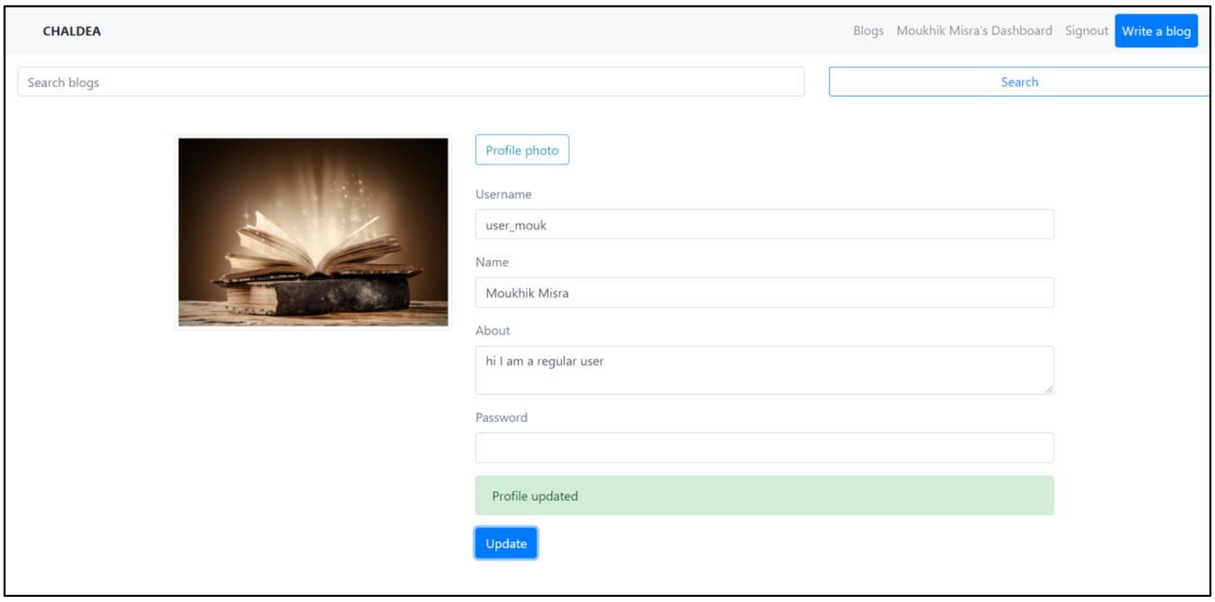
Username: user_mouk

Name: Moukhik Misra

About: hello I am a regular user

Password: ••••••

Updated Profile:



Username

user_mouk

Name

Moukhik Misra

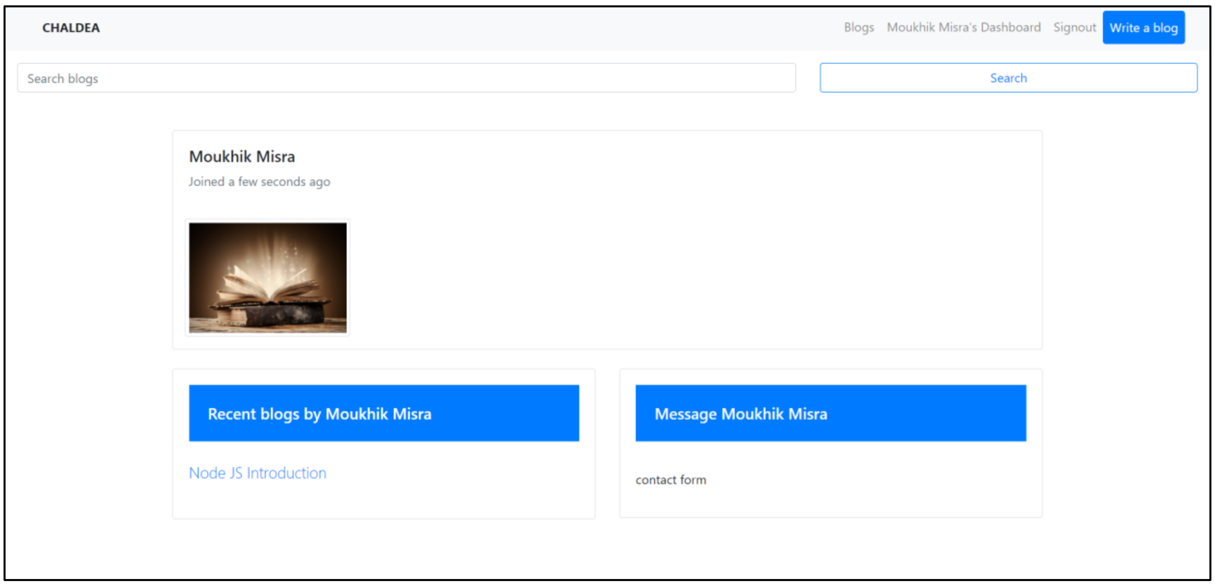
About

hi I am a regular user

Password

Profile updated

Public Profile:



Moukhik Misra

Joined a few seconds ago

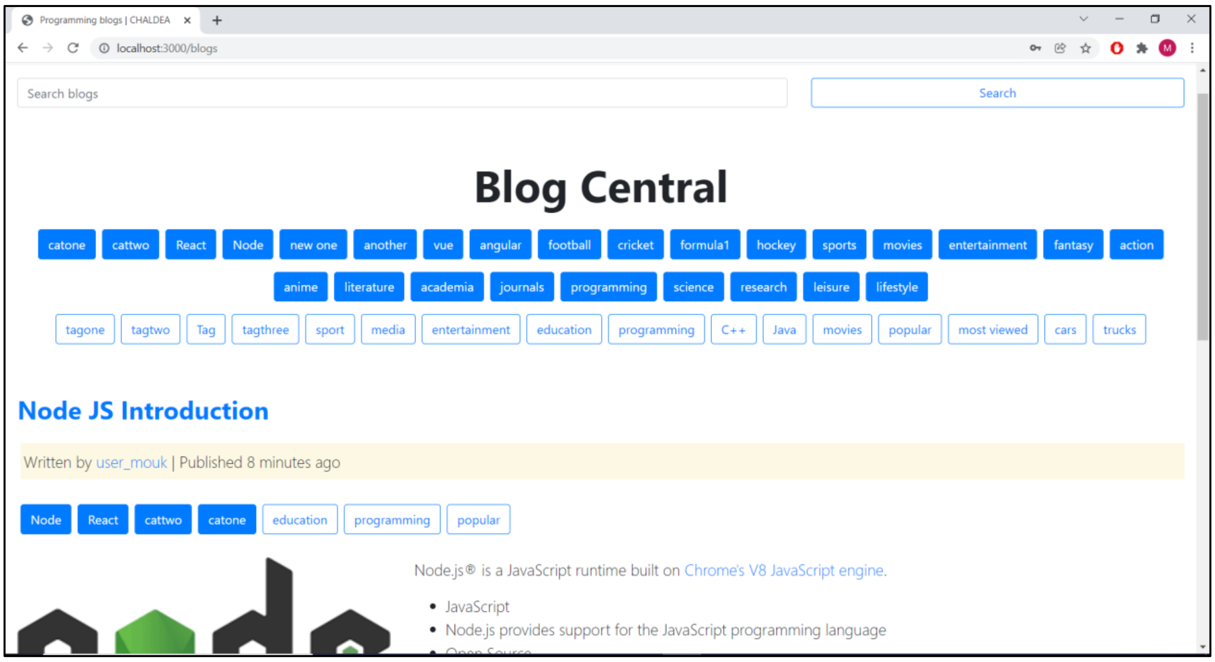
Recent blogs by Moukhik Misra

Node JS Introduction

Message Moukhik Misra

contact form

Blog Central (Displays all blogs):



Programming blogs | CHALDEA

localhost:3000/blogs

Search blogs

Search

Blog Central

catone cattwo React Node new one another vue angular football cricket formula1 hockey sports movies entertainment fantasy action

anime literature academia journals programming science research leisure lifestyle

tagone tagtwo Tag tagthree sport media entertainment education programming C++ Java movies popular most viewed cars trucks

Node JS Introduction

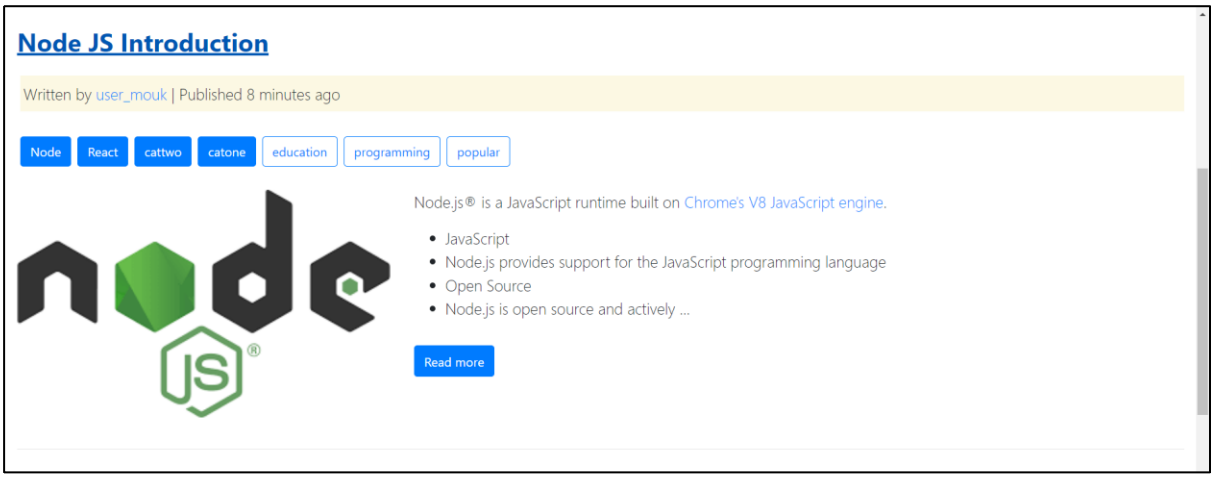
Written by user_mouk | Published 8 minutes ago

Node React cattwo catone education programming popular

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

- JavaScript
- Node.js provides support for the JavaScript programming language

(Newly Created Blog)



Node JS Introduction

Written by user_mouk | Published 8 minutes ago

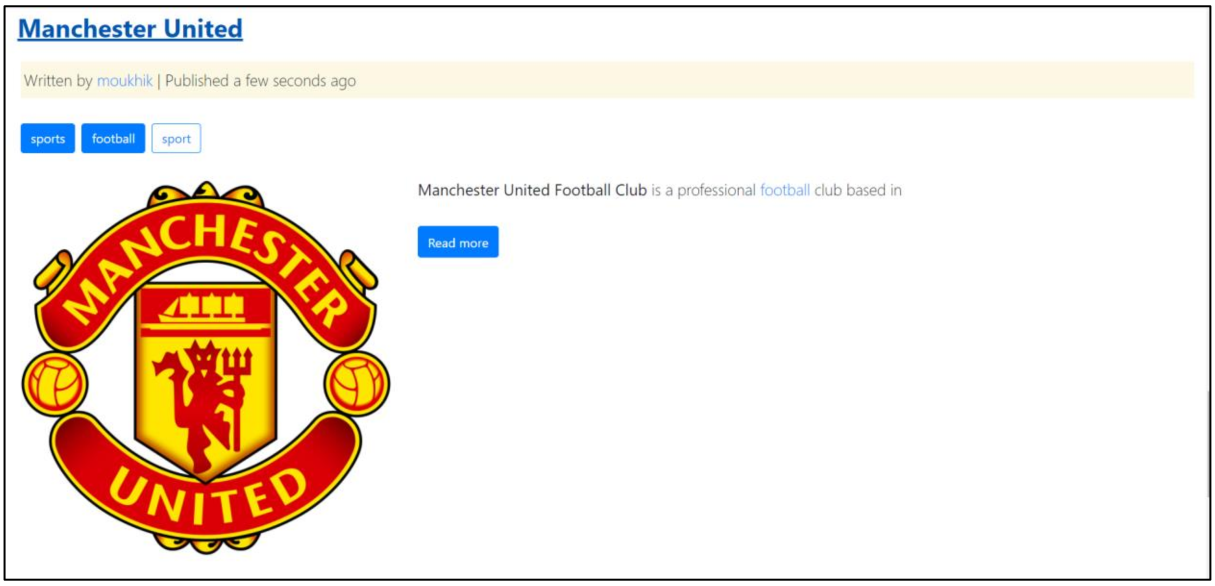
Node React cattwo catone education programming popular

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

- JavaScript
- Node.js provides support for the JavaScript programming language
- Open Source
- Node.js is open source and actively ...

Read more

(Old blogs)



Manchester United

Written by moukhik | Published a few seconds ago

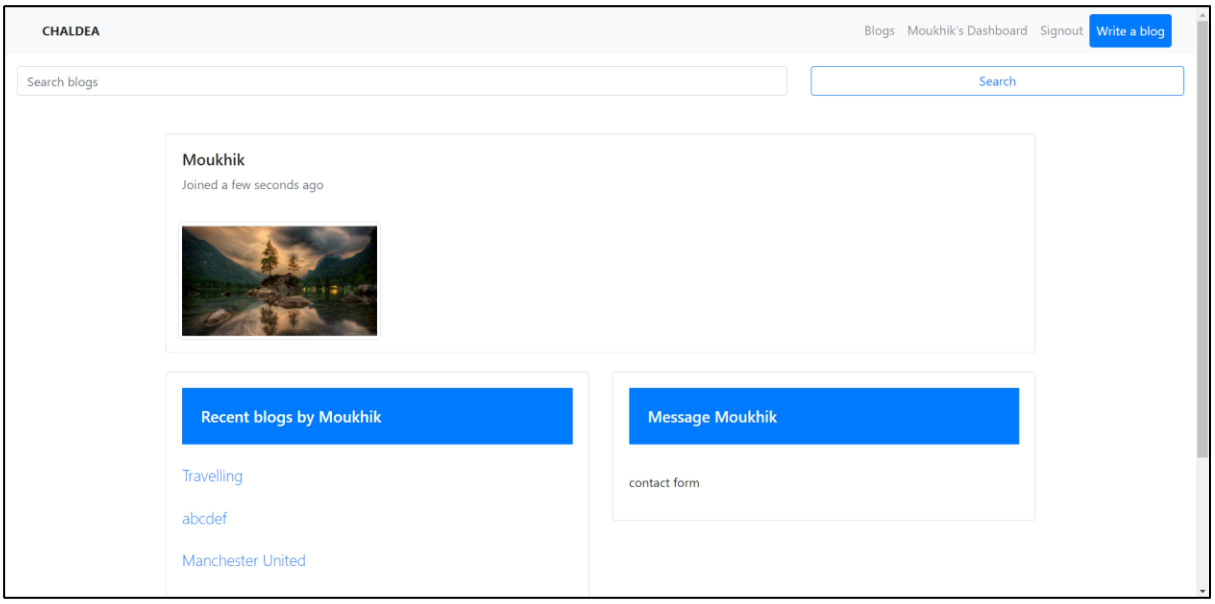
sports football sport

Manchester United Football Club is a professional football club based in

Read more

(Created by admin user moukhik)

Moukhik Public profile:



CHALDEA Blogs Moukhik's Dashboard Signout Write a blog

Search blogs Search

Moukhik

Joined a few seconds ago

Recent blogs by Moukhik

Travelling

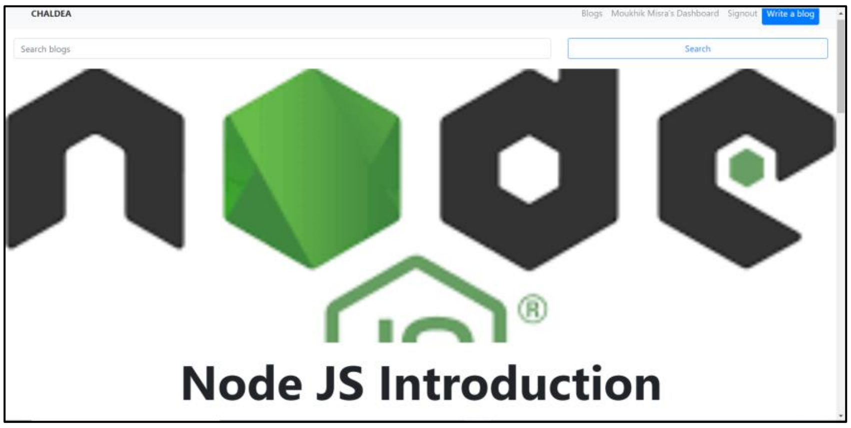
abcdef

Manchester United

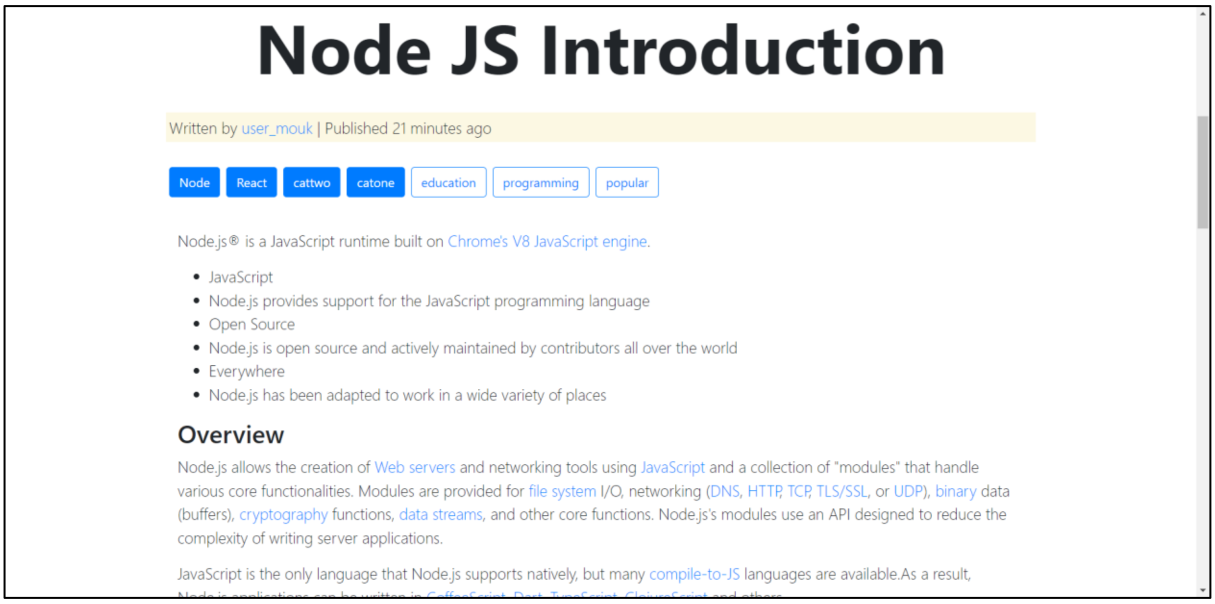
Message Moukhik

contact form

Viewing the newly created blog by user_mouk:



Node JS Introduction



Node JS Introduction

Written by user_mouk | Published 21 minutes ago

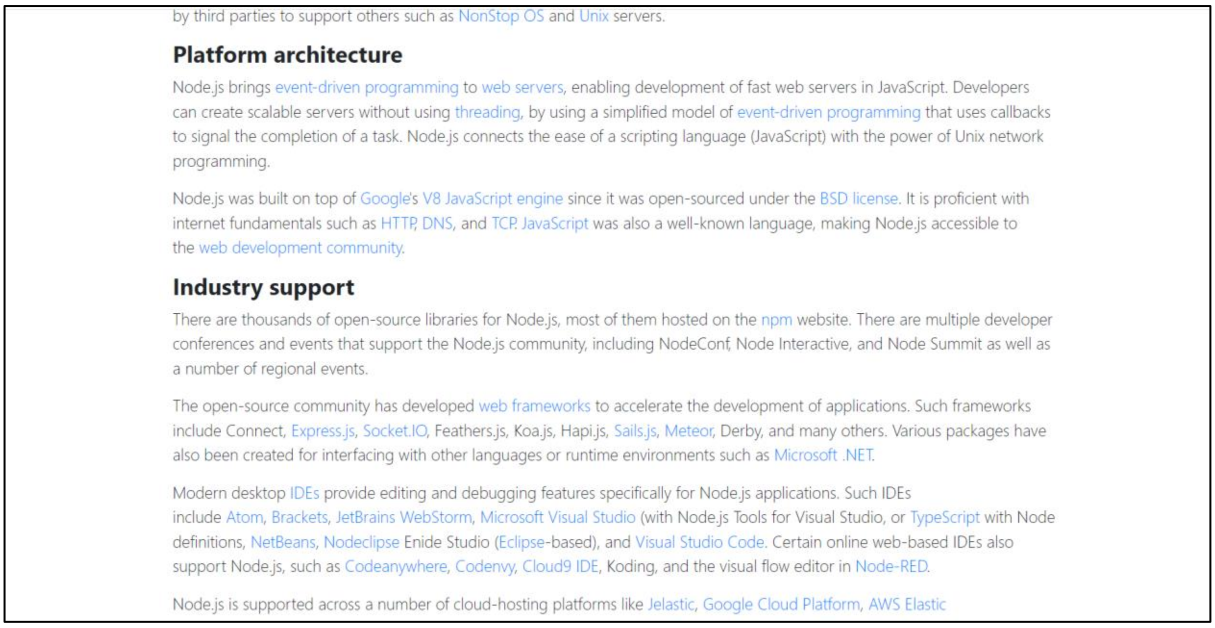
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

- JavaScript
- Node.js provides support for the JavaScript programming language
- Open Source
- Node.js is open source and actively maintained by contributors all over the world
- Everywhere
- Node.js has been adapted to work in a wide variety of places

Overview

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.

JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available.As a result, Node.js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.



by third parties to support others such as NonStop OS and Unix servers.

Platform architecture

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js connects the ease of a scripting language (JavaScript) with the power of Unix network programming.

Node.js was built on top of Google's V8 JavaScript engine since it was open-sourced under the BSD license. It is proficient with internet fundamentals such as HTTP, DNS, and TCP. JavaScript was also a well-known language, making Node.js accessible to the web development community.

Industry support

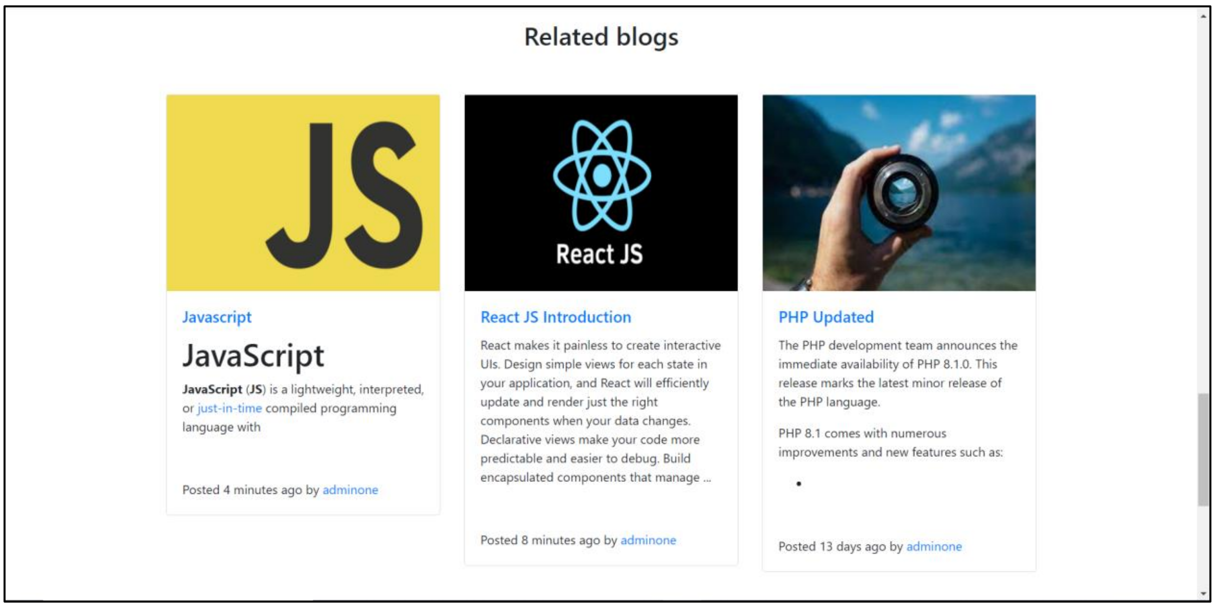
There are thousands of open-source libraries for Node.js, most of them hosted on the npm website. There are multiple developer conferences and events that support the Node.js community, including NodeConf, Node Interactive, and Node Summit as well as a number of regional events.

The open-source community has developed web frameworks to accelerate the development of applications. Such frameworks include Connect, Express.js, Socket.IO, Feathers.js, Koa.js, Hapi.js, Sails.js, Meteor, Derby, and many others. Various packages have also been created for interfacing with other languages or runtime environments such as Microsoft .NET.

Modern desktop IDEs provide editing and debugging features specifically for Node.js applications. Such IDEs include Atom, Brackets, JetBrains WebStorm, Microsoft Visual Studio (with Node.js Tools for Visual Studio, or TypeScript with Node definitions, NetBeans, Nodeclipse Enide Studio (Eclipse-based), and Visual Studio Code. Certain online web-based IDEs also support Node.js, such as Codeanywhere, Codenvy, Cloud9 IDE, Koding, and the visual flow editor in Node-RED.

Node.js is supported across a number of cloud-hosting platforms like Jelastic, Google Cloud Platform, AWS Elastic

Related Blogs (SEO Results using Categories and Tags)



Related blogs

Javascript

JavaScript

JavaScript (**JS**) is a lightweight, interpreted, or just-in-time compiled programming language with

Posted 4 minutes ago by adminone

React JS Introduction

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug. Build encapsulated components that manage ...

Posted 8 minutes ago by adminone

PHP Updated

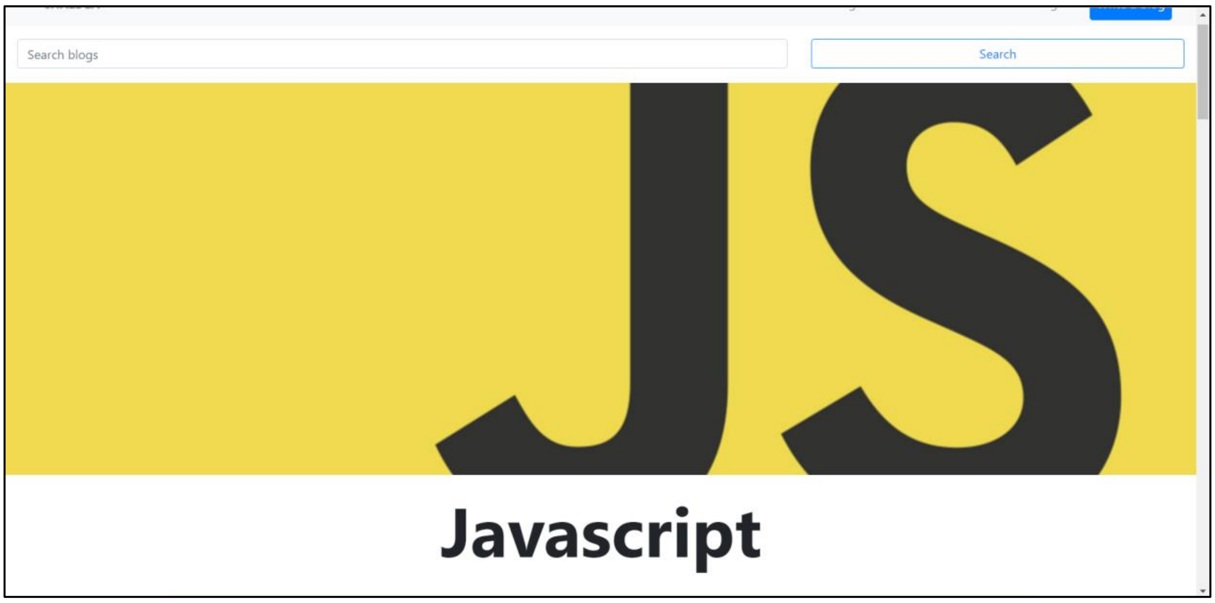
The PHP development team announces the immediate availability of PHP 8.1.0. This release marks the latest minor release of the PHP language.

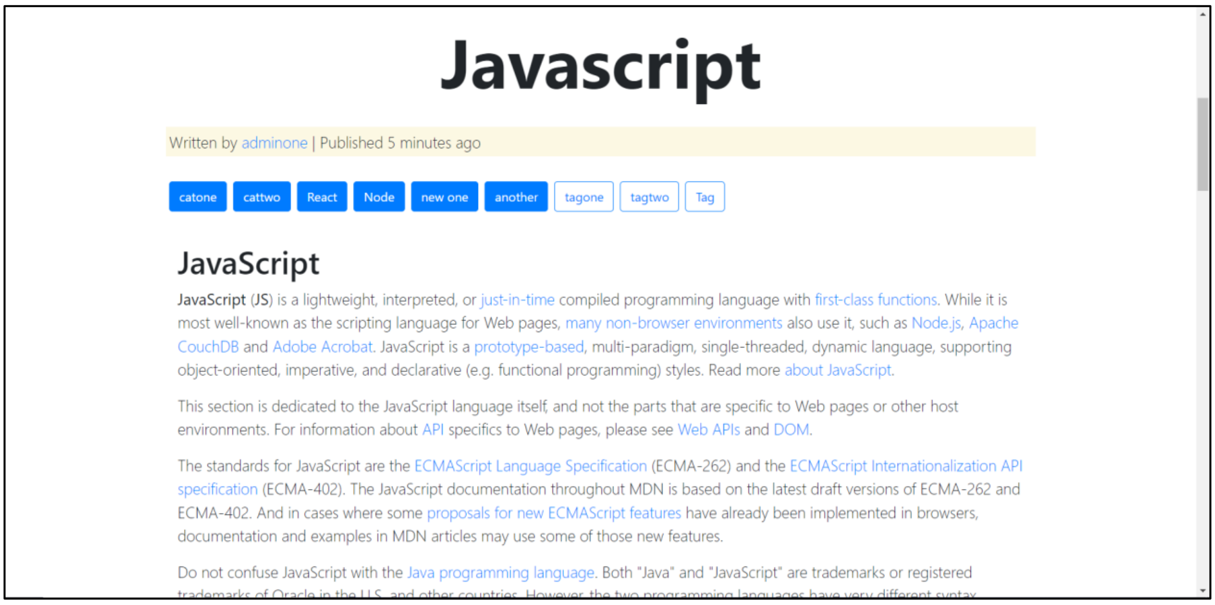
PHP 8.1 comes with numerous improvements and new features such as:

•

Posted 13 days ago by adminone

Going to JS blog





Javascript

Written by adminone | Published 5 minutes ago

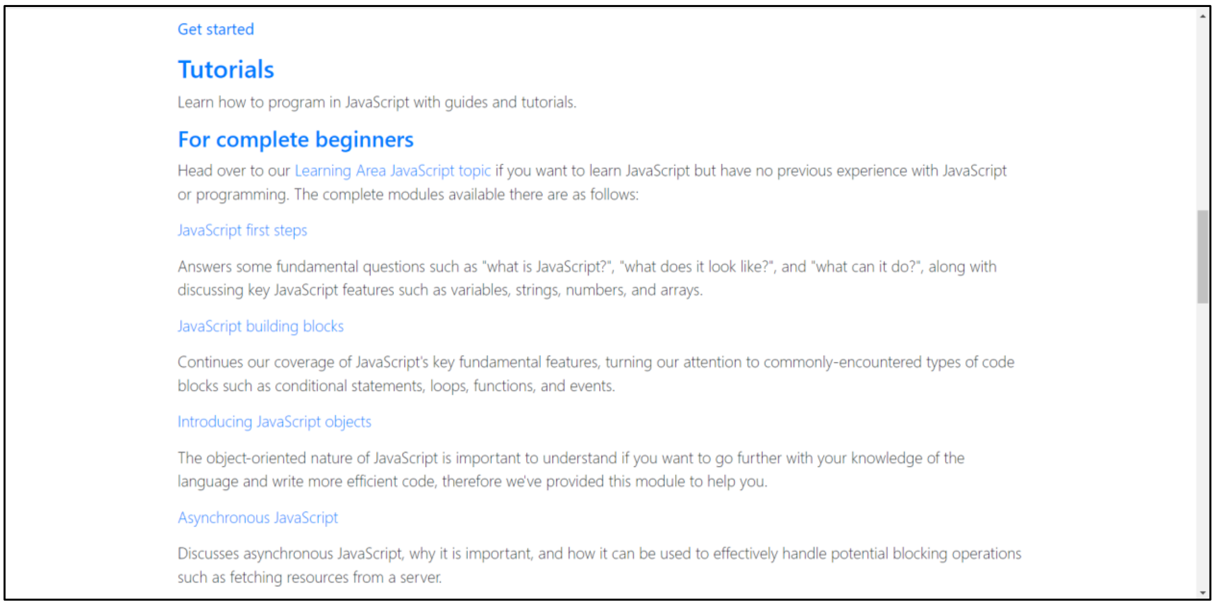
JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles. Read more about JavaScript.

This section is dedicated to the JavaScript language itself, and not the parts that are specific to Web pages or other host environments. For information about API specifics to Web pages, please see Web APIs and DOM.

The standards for JavaScript are the ECMAScript Language Specification (ECMA-262) and the ECMAScript Internationalization API specification (ECMA-402). The JavaScript documentation throughout MDN is based on the latest draft versions of ECMA-262 and ECMA-402. And in cases where some proposals for new ECMAScript features have already been implemented in browsers, documentation and examples in MDN articles may use some of those new features.

Do not confuse JavaScript with the Java programming language. Both "Java" and "JavaScript" are trademarks or registered



Get started

Tutorials

Learn how to program in JavaScript with guides and tutorials.

For complete beginners

Head over to our Learning Area JavaScript topic if you want to learn JavaScript but have no previous experience with JavaScript or programming. The complete modules available there are as follows:

JavaScript first steps

Answers some fundamental questions such as "what is JavaScript?", "what does it look like?", and "what can it do?", along with discussing key JavaScript features such as variables, strings, numbers, and arrays.

JavaScript building blocks

Continues our coverage of JavaScript's key fundamental features, turning our attention to commonly-encountered types of code blocks such as conditional statements, loops, functions, and events.

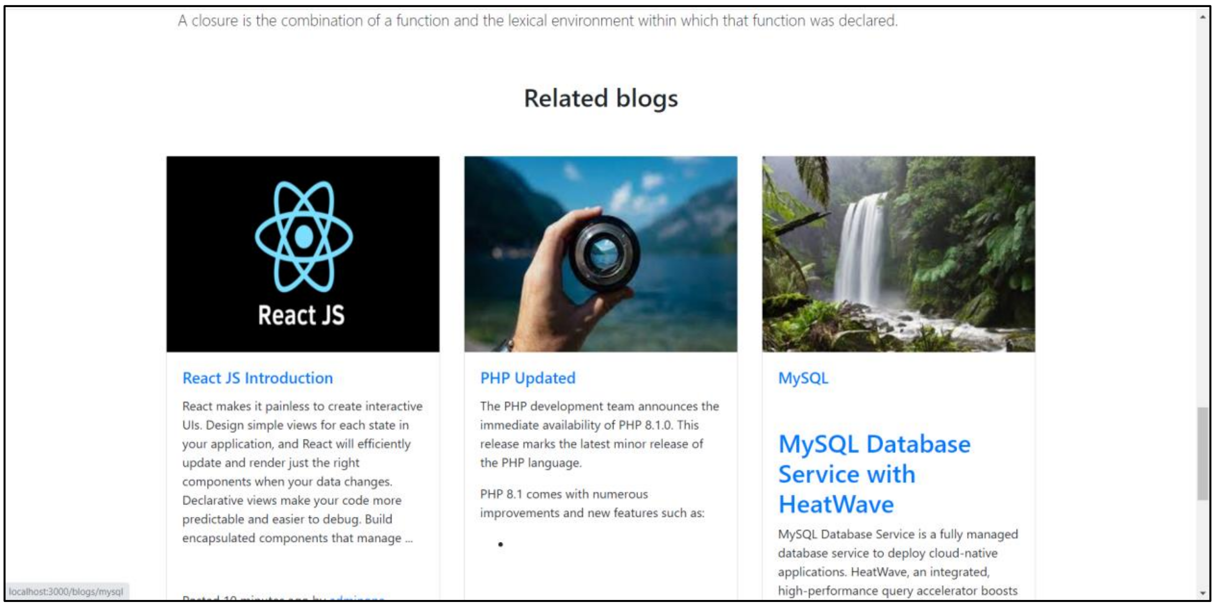
Introducing JavaScript objects

The object-oriented nature of JavaScript is important to understand if you want to go further with your knowledge of the language and write more efficient code, therefore we've provided this module to help you.

Asynchronous JavaScript

Discusses asynchronous JavaScript, why it is important, and how it can be used to effectively handle potential blocking operations such as fetching resources from a server.

SEO Related Blogs for JS Blog



A closure is the combination of a function and the lexical environment within which that function was declared.

Related blogs

React JS Introduction

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug. Build encapsulated components that manage ...

PHP Updated

The PHP development team announces the immediate availability of PHP 8.1.0. This release marks the latest minor release of the PHP language.

PHP 8.1 comes with numerous improvements and new features such as:

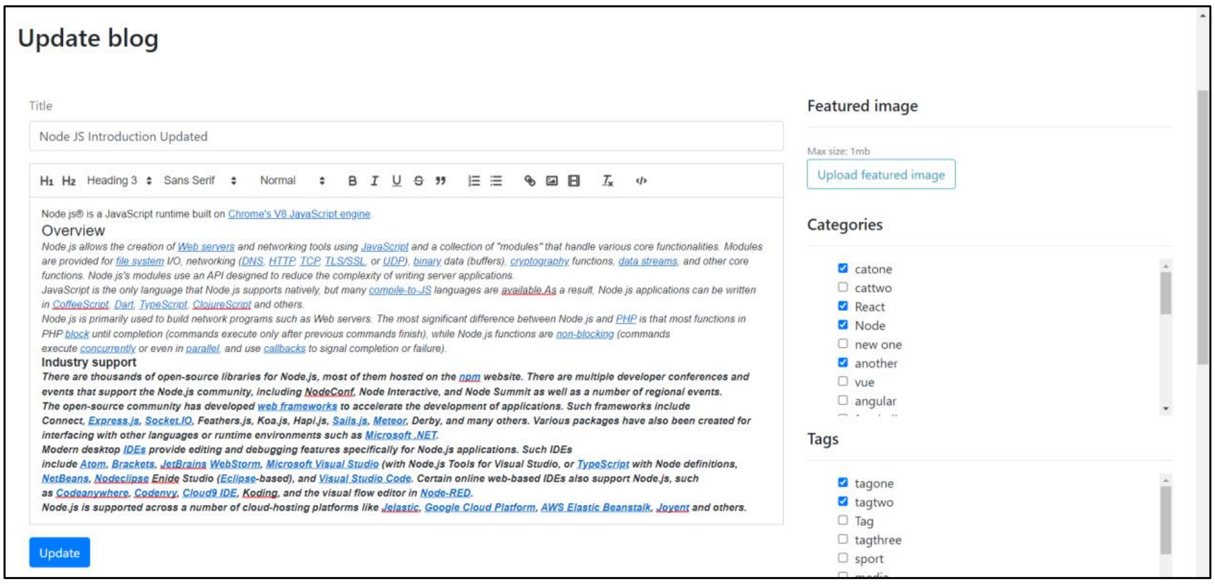
-

MySQL

MySQL Database Service with HeatWave

MySQL Database Service is a fully managed database service to deploy cloud-native applications. HeatWave, an integrated, high-performance query accelerator boosts

Update Blog (Updating newly created blogs)


Update blog

Title

Node JS Introduction Updated

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Overview

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.
JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available.As a result, Node.js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.
Node.js is primarily used to build network programs such as Web servers. The most significant difference between Node.js and PHP is that most functions in PHP block until completion (commands execute only after previous commands finish), while Node.js functions are non-blocking (commands execute concurrently or even in parallel, and use callbacks to signal completion or failure).

Industry support

There are thousands of open-source libraries for Node.js, most of them hosted on the npm website. There are multiple developer conferences and events that support the Node.js community, including NodeConf, Node Interactive, and Node Summit as well as a number of regional events.
The open-source community has developed web frameworks to accelerate the development of applications. Such frameworks include Connect, Express.js, Socket.IO, Feathers.js, Koa.js, Hapi.js, Sails.js, Meteor, Derby, and many others. Various packages have also been created for interfacing with other languages or runtime environments such as Microsoft .NET.
Modern desktop IDEs provide editing and debugging features specifically for Node.js applications. Such IDEs include Atom, Brackets, JetBrains WebStorm, Microsoft Visual Studio (with Node.js Tools for Visual Studio, or TypeScript with Node definitions, NetBeans, Nodeclipse Enide Studio (Eclipse-based), and Visual Studio Code. Certain online web-based IDEs also support Node.js, such as Codeanywhere, Codenvy, Cloud9 IDE, Koding, and the visual flow editor in Node-RED.
Node.js is supported across a number of cloud-hosting platforms like Jelastic, Google Cloud Platform, AWS Elastic Beanstalk, Joyent and others.

Featured image

Max size: 1mb

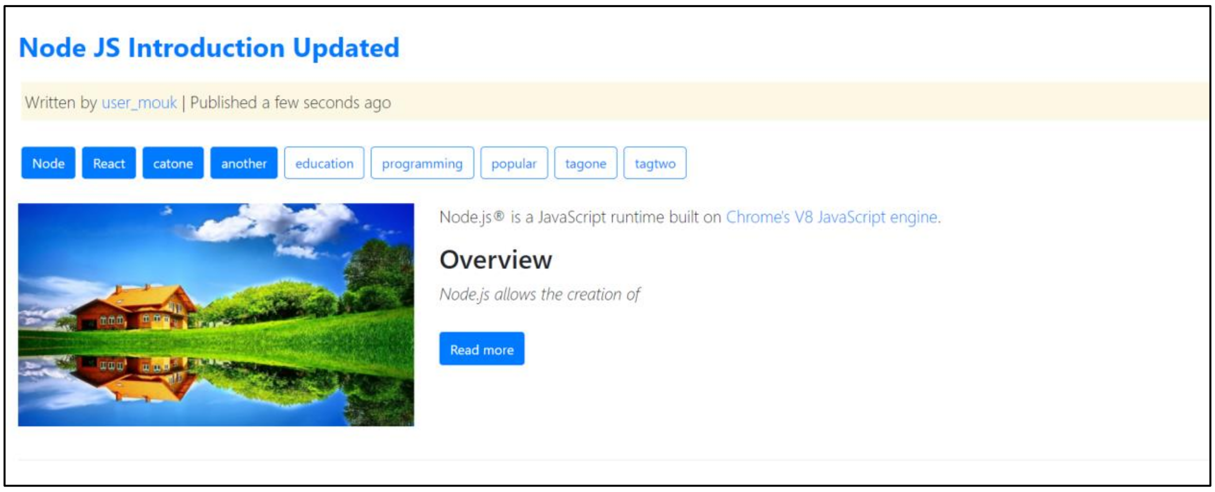
Categories

- [✓] catone
- [ ] cattwo
- [✓] React
- [✓] Node
- [ ] new one
- [✓] another
- [ ] vue
- [ ] angular

Tags

- [✓] tagone
- [✓] tagtwo
- [ ] Tag
- [ ] tagthree
- [ ] sport

Updated on Blog Central

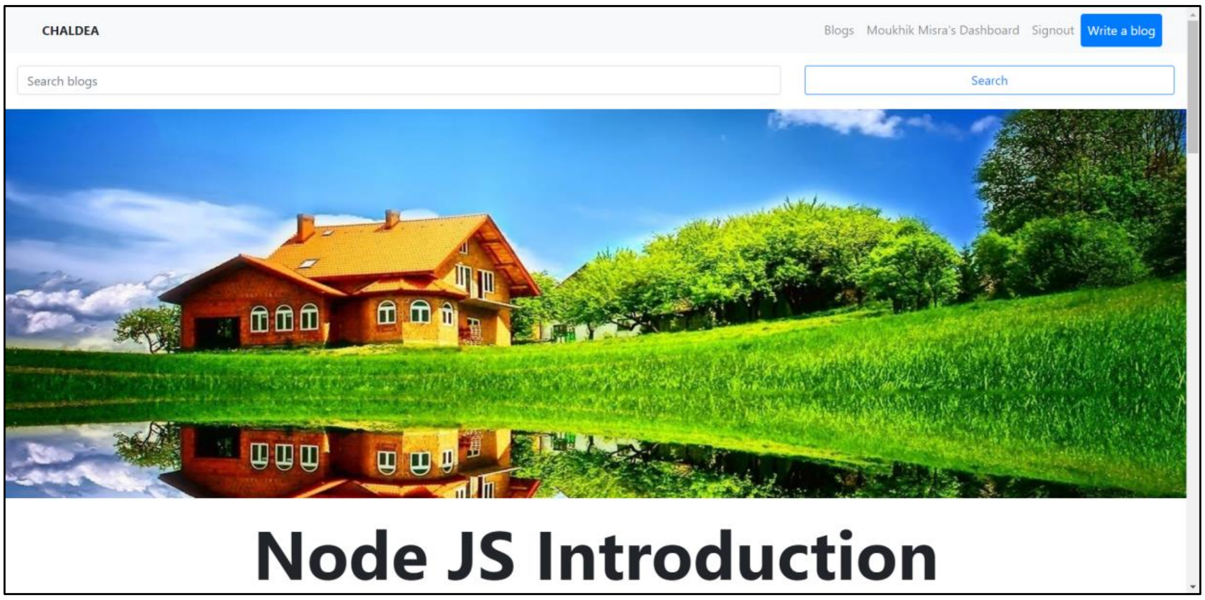

Node JS Introduction Updated

Written by user_mouk | Published a few seconds ago

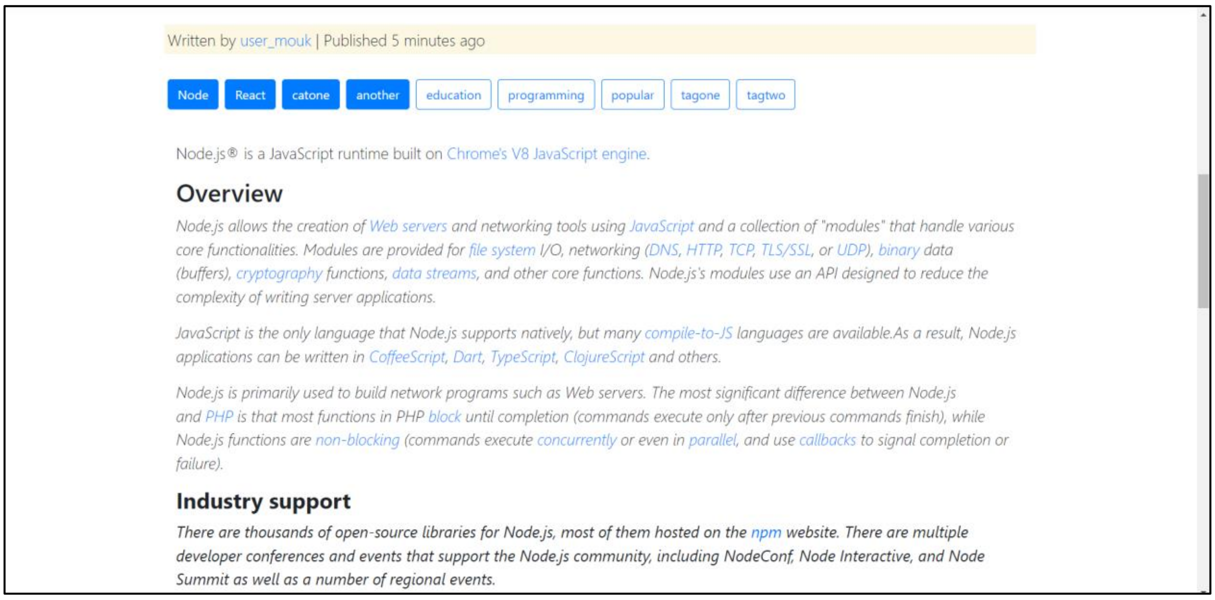
Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Overview

Node.js allows the creation of



Node JS Introduction


Written by user_mouk | Published 5 minutes ago

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Overview

Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities. Modules are provided for file system I/O, networking (DNS, HTTP, TCP, TLS/SSL, or UDP), binary data (buffers), cryptography functions, data streams, and other core functions. Node.js's modules use an API designed to reduce the complexity of writing server applications.

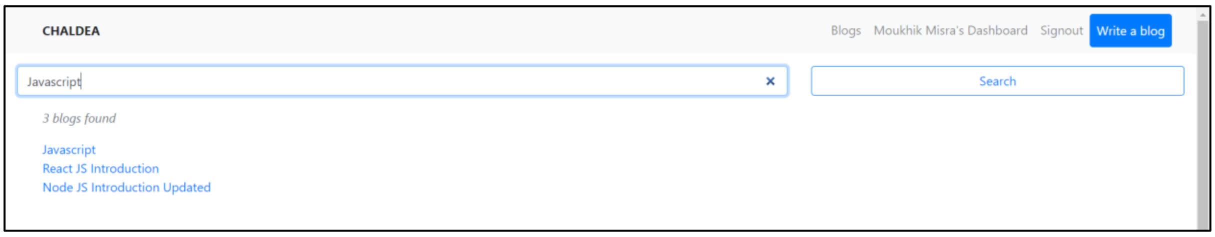
JavaScript is the only language that Node.js supports natively, but many compile-to-JS languages are available.As a result, Node.js applications can be written in CoffeeScript, Dart, TypeScript, ClojureScript and others.

Node.js is primarily used to build network programs such as Web servers. The most significant difference between Node.js and PHP is that most functions in PHP block until completion (commands execute only after previous commands finish), while Node.js functions are non-blocking (commands execute concurrently or even in parallel, and use callbacks to signal completion or failure).

Industry support

There are thousands of open-source libraries for Node.js, most of them hosted on the npm website. There are multiple developer conferences and events that support the Node.js community, including NodeConf, Node Interactive, and Node Summit as well as a number of regional events.

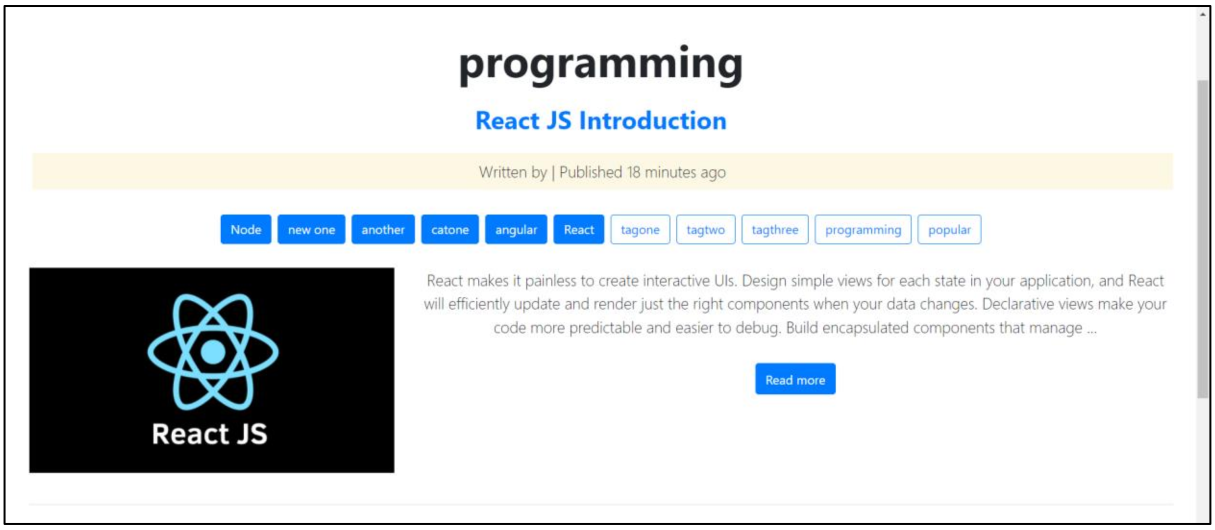
Search Functionality:



3 blogs found

Javascript
React JS Introduction
Node JS Introduction Updated

Displaying all blogs of a particular tag (programming tag)

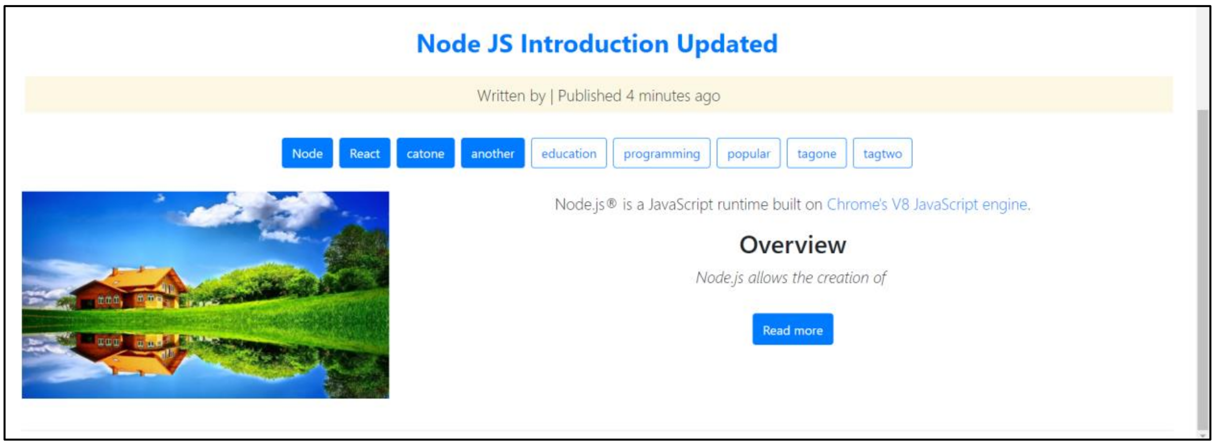


programming

React JS Introduction

Written by | Published 18 minutes ago

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug. Build encapsulated components that manage ...



Node JS Introduction Updated

Written by | Published 4 minutes ago

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Overview

Node.js allows the creation of

CONCLUSION

In this project we successfully created a content management platform for blog creation with pictures and dedicated formatting tools for users. We made a login and signup module for existing and new users. The website had search engine optimisation implemented allowing users to browse through the content within the site by entering relevant keywords in the search bar that shows search results based on blog categories and associated tags. Our webpage allows users to perform all the basic creation, updation and deletion operations on their blogs. We created an admin component to the project where the admin is redirected to pages with exclusive admin capabilities of adding or deleting blogs, tags and categories for all users. The routes were protected by means of a token-based user role allocation for access control over admin functionalities. The sensitive data is stored in the database after being hashed making it even more secure. We successfully used Quill which is a react component to make a rich text editor for better user experience. The blog hub displays the images uploaded by authors along with excerpts from their articles for other users to discover them. The user account has a display name and profile picture and can create a public profile to let other users see blogs posted by them. The additional features that may be added to our project in the future include and are not limited to; email functionality for contacting blog authors, deep learning algorithms to recognise user content consumption patterns and make more relevant suggestions, and more responsive user interface that can be deployed on a large scale.

REFERENCES

[1] A. Tiwari and S. Chaturvedi, “Optimized technique for ranking webpage on search engine optimization,” Proc. - 2nd Int. Conf. Micro-Electronics Telecommun. Eng. ICMETE 2018, no. 2016, pp. 107–110, 2018, doi: 10.1109/ICMETE.2018.00034.

[2] L. H. Ho, M. H. Lu, J. C. Huang, and H. Y. Ho, “The application of search engine optimization for internet marketing: An example of the motel websites,” 2010 2nd Int. Conf. Comput. Autom. Eng. ICCAE 2010, vol. 1, pp. 380–383, 2010, doi: 10.1109/ICCAE.2010.5451929.

[3] J. Y. Lemos and A. R. Joshi, “Search engine optimization to enhance user interaction,” Proc. Int. Conf. IoT Soc. Mobile, Anal. Cloud, I-SMAC 2017, pp. 398–402, 2017, doi: 10.1109/I-SMAC.2017.8058379.

[4] Y. Mo, “A study on tactics for corporate website development aiming at search engine optimization,” 2nd Int. Work. Educ. Technol. Comput. Sci. ETCS 2010, vol. 3, pp. 673–675, 2010, doi: 10.1109/ETCS.2010.230.

[5] H. Zhou, S. Qin, J. Liu, and J. Chen, “Study on website search engine optimization,” Proc. - 2012 Int. Conf. Comput. Sci. Serv. Syst. CSSS 2012, pp. 930–933, 2012, doi: 10.1109/CSSS.2012.236.