A Major project report on

**PUBLIC BLOG APPLICATION USING MERN STACK**

Submitted to P. B. Siddhartha College of Arts & Science for the partial fulfillment of the requirements for the award of

**Degree of Bachelor Science (Bsc-CAMS)**

**By**

**T.Manasa ( 203661)**

**G.Renuka(203665)**

**L.Leela Vamsi(203617)**

Under the esteemed guidance of

Mrs V.Jhansi



DEPARTMENT OF COMPUTER SCIENCE

P. B. SIDDHARTHA COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)

Vijayawada – 520010

**P. B. SIDDHARTHA COLLEGE OF ARTS AND SCIENCE (AUTONOMOUS)**

**Vijayawada – 520010**



**DEPARTMENT OF COMPUTER SCIENCE**

CERTIFICATE

This is certified that project done in Computer science entitled with **“Public blog application using MERN STACK”**done by **T.Manasa(203661)** for the partial fulfillment of the requirement for the award of degree Bachelor of Science and Arts as part of the curriculum during the academic year 2021 - 2023.

Date:

Lecturer in charge Head of the Department

**ACKNOWLEDGEMENT**

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

First of all I would like to express my profound sense of gratitude to the Principal **Dr. M. V. N Padma Rao** of or giving me the permission to carry out this project.

I express my deep gratitude and regards to **Mr. K. Sridhar**, Head of the Department of Computer Science, and project guide, **Mrs.V.Jhansi**, Lecturer, Department in Computer Science for their encouragement and valuable guidance in bringing shape to this dissertation.

I am thankful to all the Technical Staff and Non-teaching staff in the department for their teachings and academic support.

**T.MANASA(203661)**

**G. RENUKA(203665)**

**L.LEELA VAMSI(203617)**

**PUBLIC BLOG APPLICATION**

**Abstract:**

Blogging is as simple as obtaining a website and publishing original content on it. Tech-savvy bloggers can buy a domain name and build the website themselves. Those with less HTML knowledge can create an account with sites like WordPress that simplify the web design and publishing process.Blogs are usually simple websites. Older pieces may be archived in separate sections of the site, and there may be a separate page with contact info or a bio, but the blog itself is usually just a single page that can be scrolled through—similar to the news feed on social media sites like Facebook. As with a Facebook news feed, a blog displays the newest content at the top of the page.

Another unique feature of blogging is interlinking. This occurs when a blogger links to another person's blog within their own blog post. For example, if a music teacher maintains a blog, and they write a blog post about how to form a chord, they might link to a musician's blog to show an example of the chords in action. A political blogger may link to another politics blog and then discuss how they agree or disagree with a post on that blog. Interlinking, along with the comment section, fosters the sense of community that makes blogs unique.The purpose of online blogging system is to automate the existing manual syatem by the help of computarized equipments and full-fledged computer software, fulfilling their requirements, so that their valuabale data/information can be stored for a longer period of time with easy accessing and manipulation of the same. The required software and hardware are easily availbale and easy to work with.A blog, short for weblog, is a frequently updated web page used for **personal commentary or business content**. Blogs are often interactive and include sections at the bottom of individual blog posts where readers can leave comments.

Blogs have evolved dramatically in recent years and many have become central forums for public conversation and even journalism. Vanity Blogs, which are purely an individual's personal pulpit may, in most cases, be easily excluded from an encyclopedic record, but open multi-user multi-threading forums are an entirely different case. These forums have become more a Public Square than has ever existed and will, as a class, only grow in significance. Even as a matter of simple historic record, the significant multi-user blogs of today will be items of interest for future research, and a concise record and history of them should be a part of a project like Wikipedia.

**SYSTEM ANALYSIS**

**Existing system**

* There is an old and traditional system that is even followed even now,which is manual blog application system.Later on ,we started to use some tools like google documents and google sheets.Then came as previously said the biometric system of blog.Each had it owns purpose as the sole purpose off manufacturing something is to fullfill its purpose.
* As far as considering the executing systems they have served their purpose to their full extent.

**Proposed System**

* The current system is what that proves the difference in technology and time.
* As introduced earlier,this sytem captures the face and recongnizes them on comparing with the pictures in database.if there is a question of accuracy recognition ,it is achevied by continuous testing of the facial recognition system using sample data and the software used for that was Selenium IDE which we shall discuss in future.

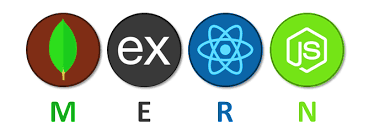
**NEED AND MOTIVATION:**

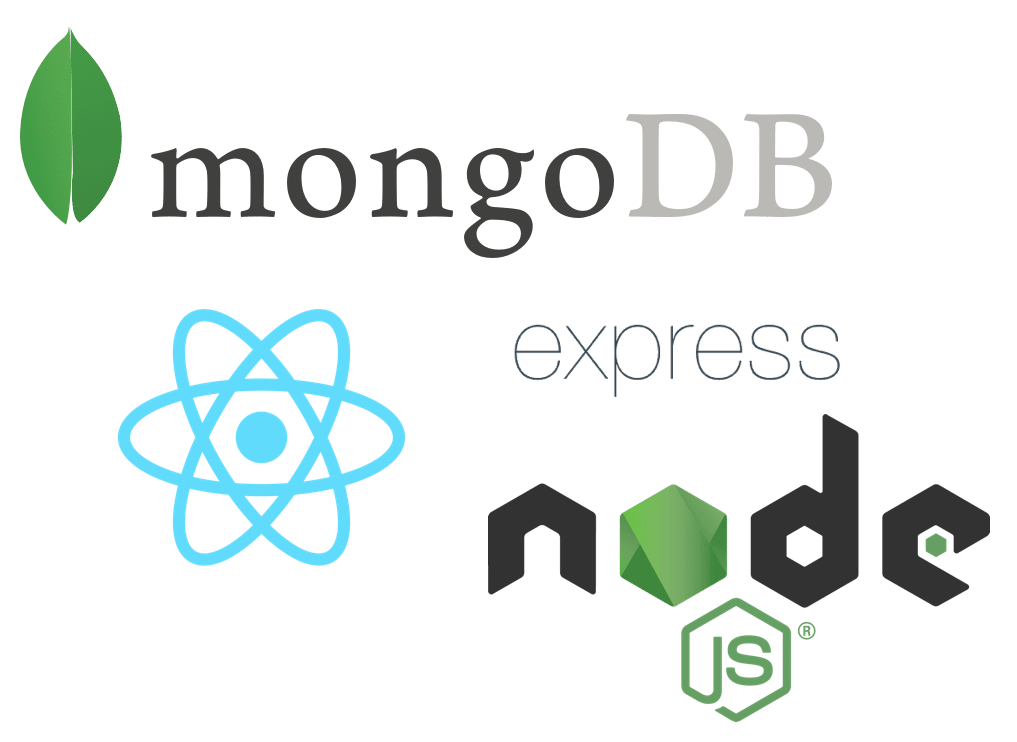
Here’s a list of some of those things blogging can help with:

1. **If you’re a business owner, it’s a great way to connect with customers**. Forget Twitter, networking events, and call centers. Blogging is a personal, down-to-earth method of keeping your customers informed and in-the-know about not only your latest product offerings, but your internal culture as well.
2. **Finding new clients.**In the same vein, don’t discount the [marketing advantages of blogging](https://www.lifehack.org/articles/money/the-1-reason-why-most-blogs-and-businesses-fail-and-the-3-questions-you-need-to-answer-to-save-yours.html). If you do it right, you could be on to something. Many businesses chalk up a large percentage of their revenue from blogging and blogging-related activities, and you can have a piece of that pie.
3. **Getting more done.**Just because you’re writing every day on a blog doesn’t mean [other things won’t get done](https://www.lifehack.org/articles/communication/how-important-is-email.html). Blogging is an activity that can literally happen *anywhere.* Wake up early, go to bed late, whatever–blogging doesn’t usually take long, and you can press writing that first sentence–you’ll find yourself invigorated, energized, and motivated by the words you’re writing.

**TOOLS AND TECHNOLIGIES USED:**

* Sublime text
* Github
* Backend-Express,Node.js,MongoDB
* Frontend-Bootstrap,React,Java script,CSS





**REQUIREMENT ANALYSIS**

FUNCTIONAL vs NONFUCTIONAL REQUIREMENTS

Funtional Requirements Nonfuntional Requirements

Objective What the product does How the product works

Focus Focus on user requirements Focus on user expectations

Documentation in use case as a quality attribute

End Result Product featres Product properties

Essentiality Mandatory Not mandatory, but desirable

Origin type Defined by user Defined by developers or tech experts

Testing Component, API, UI testing

* Functional requirements define what a product must do, what its features and functions are.
* Nonfunctional requirements describe the general properties of a system. They are also known as *quality attributes*.

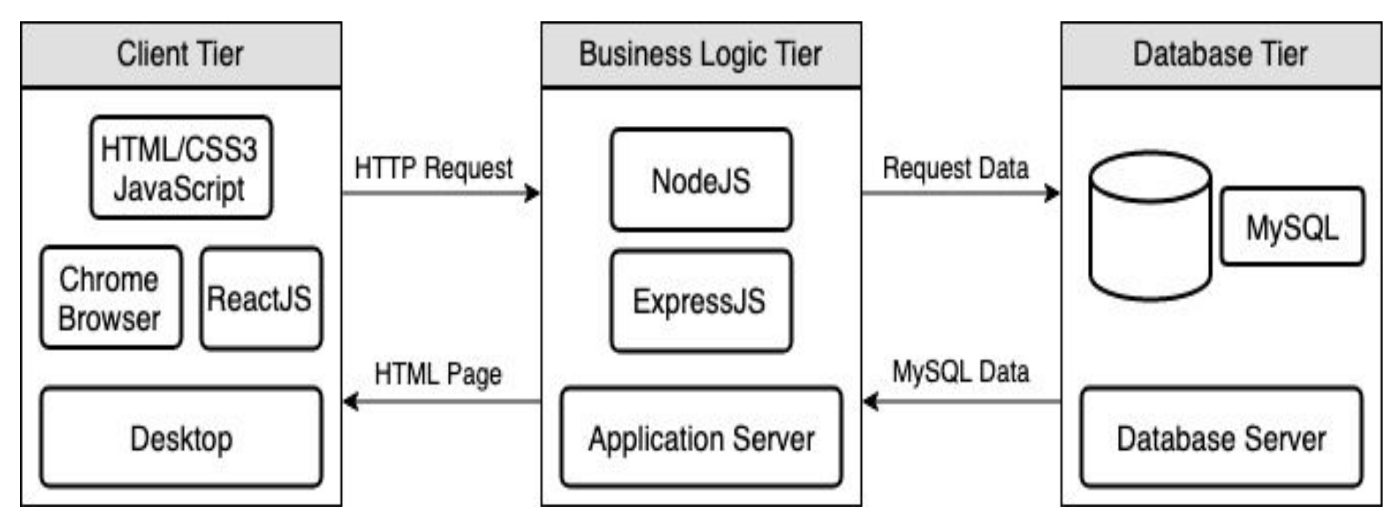
**Functional requirements** are product features or functions that developers must implement to enable users to accomplish their tasks. So, it’s important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions. For example:

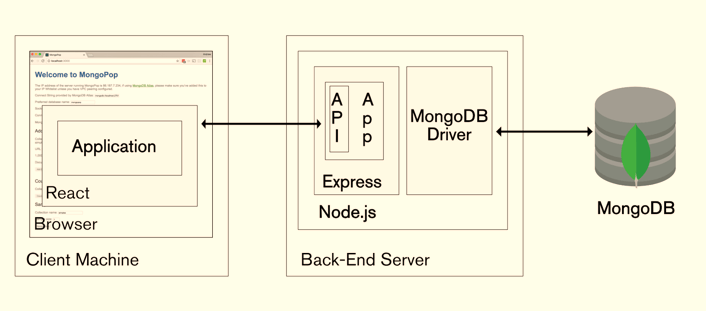
The system sends an approval request after the user enters personal information.

A search feature allows a user to hunt among various invoices if they want to credit an issued invoice.

The system sends a confirmation email when a new user account is created.

**SYSTEM DESIGN**





**IMPLEMENTATION:**

A blog, short for weblog, is a frequently updated [web page](https://www.techtarget.com/whatis/definition/page) used for personal commentary or business content. Blogs are often interactive and include sections at the bottom of individual blog posts where readers can leave comments.

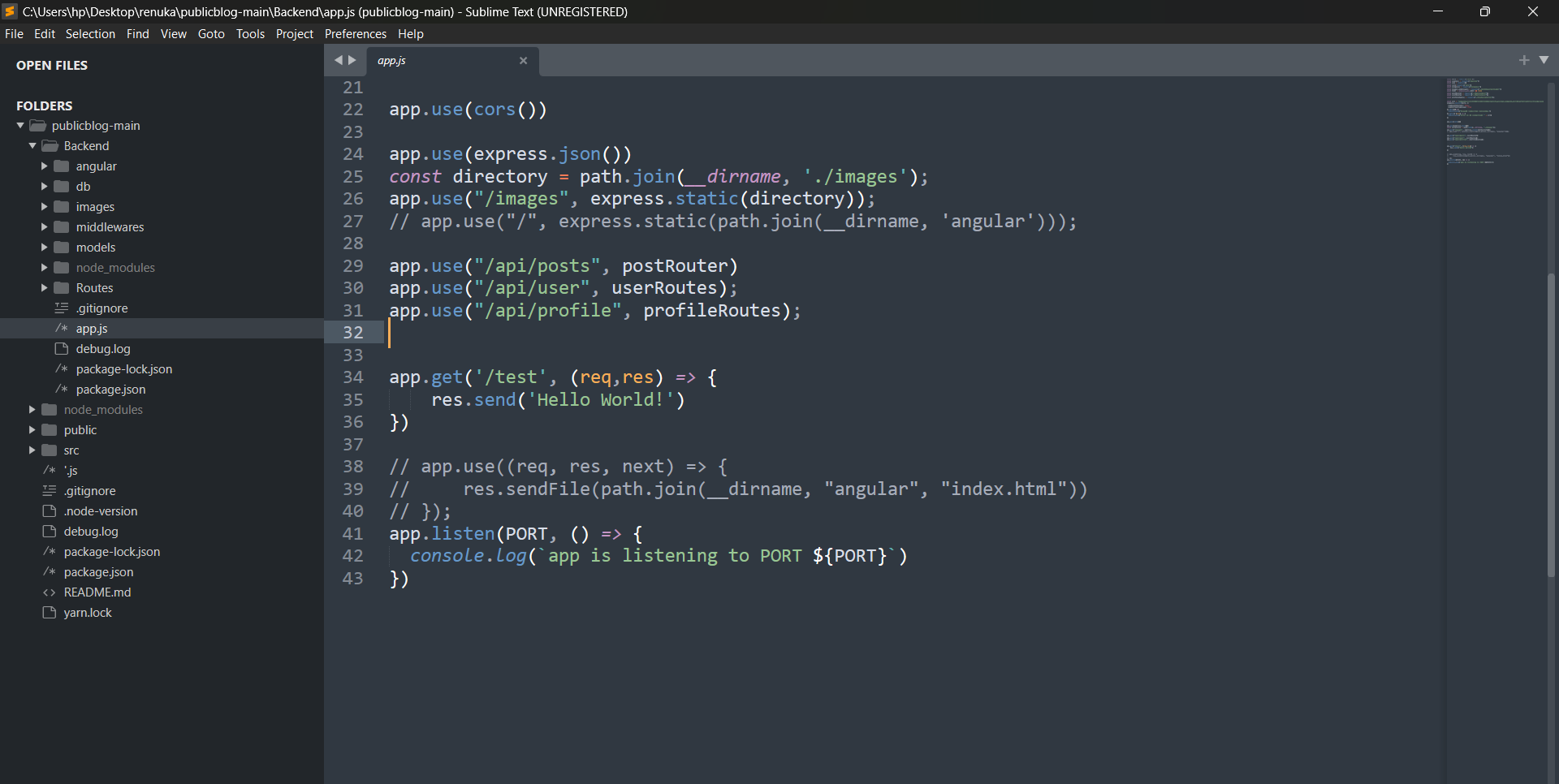
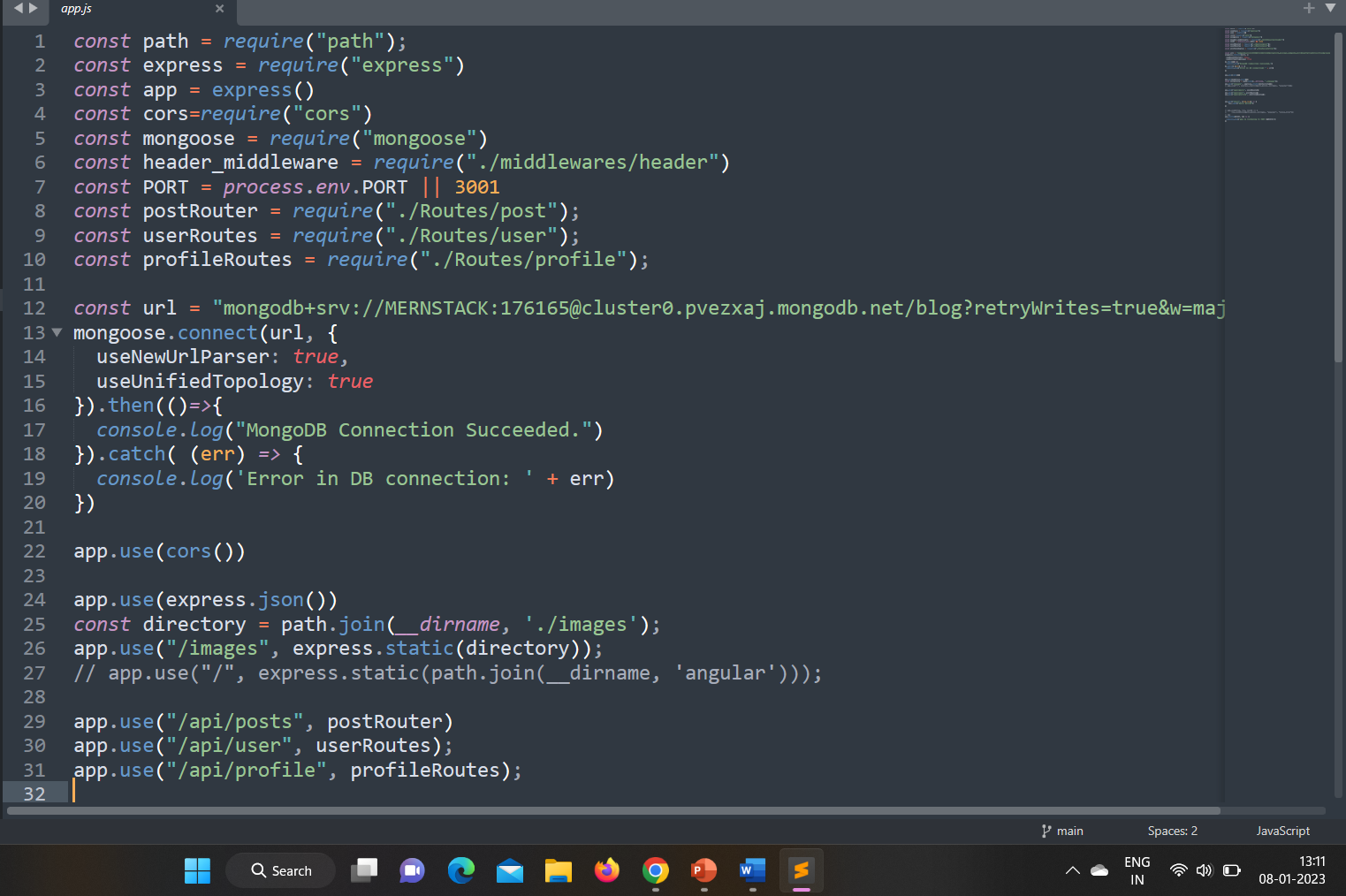
Most are written in a conversational style to reflect the voice and personal views of the blogger. Some businesses use blogs to connect with target audiences and sell products.

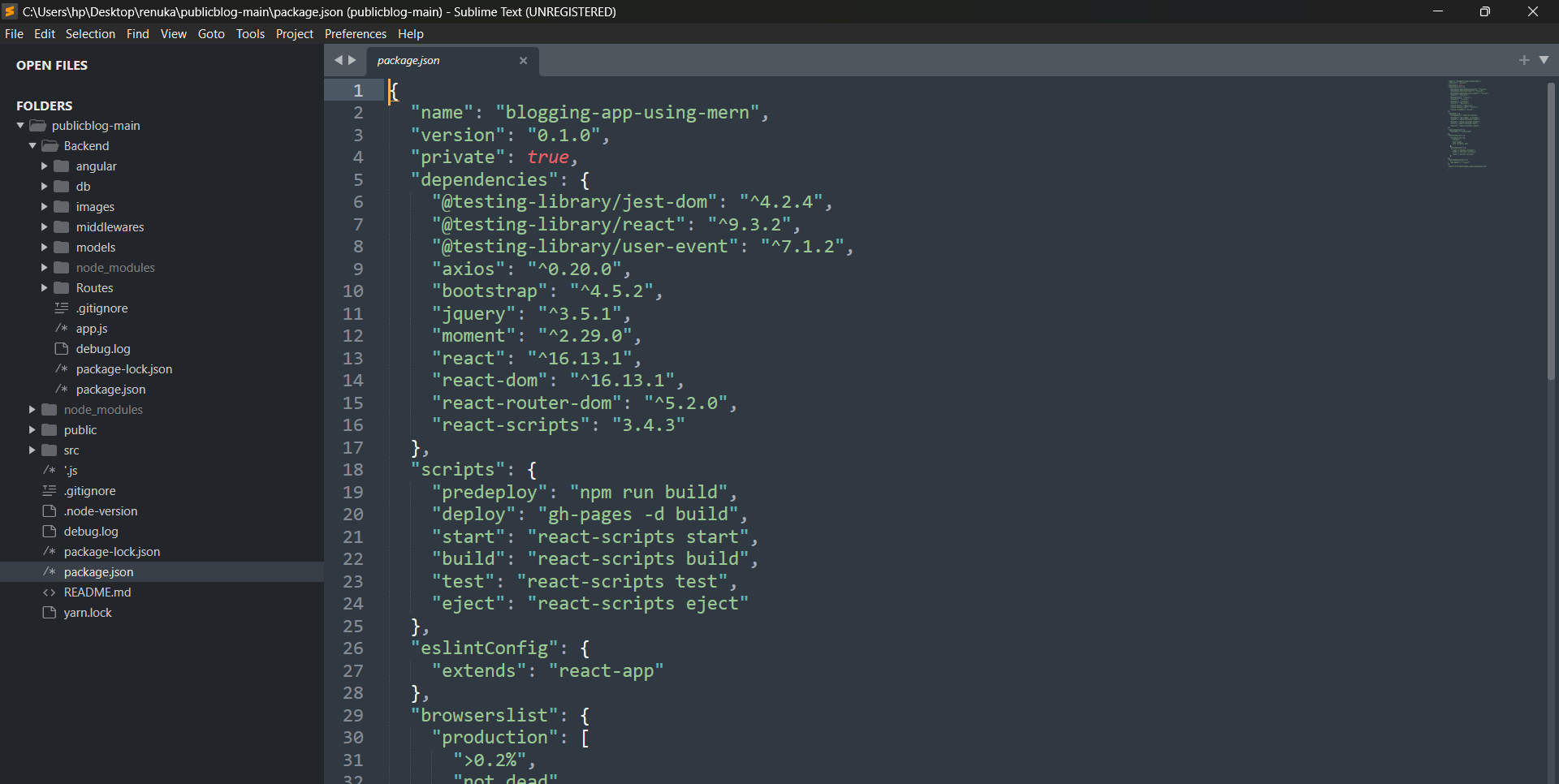
Blogs were originally called weblogs, which were websites that consisted of a series of entries arranged in reverse chronological order, so the newest posts appeared at the top. They were frequently updated with new information about various topics.

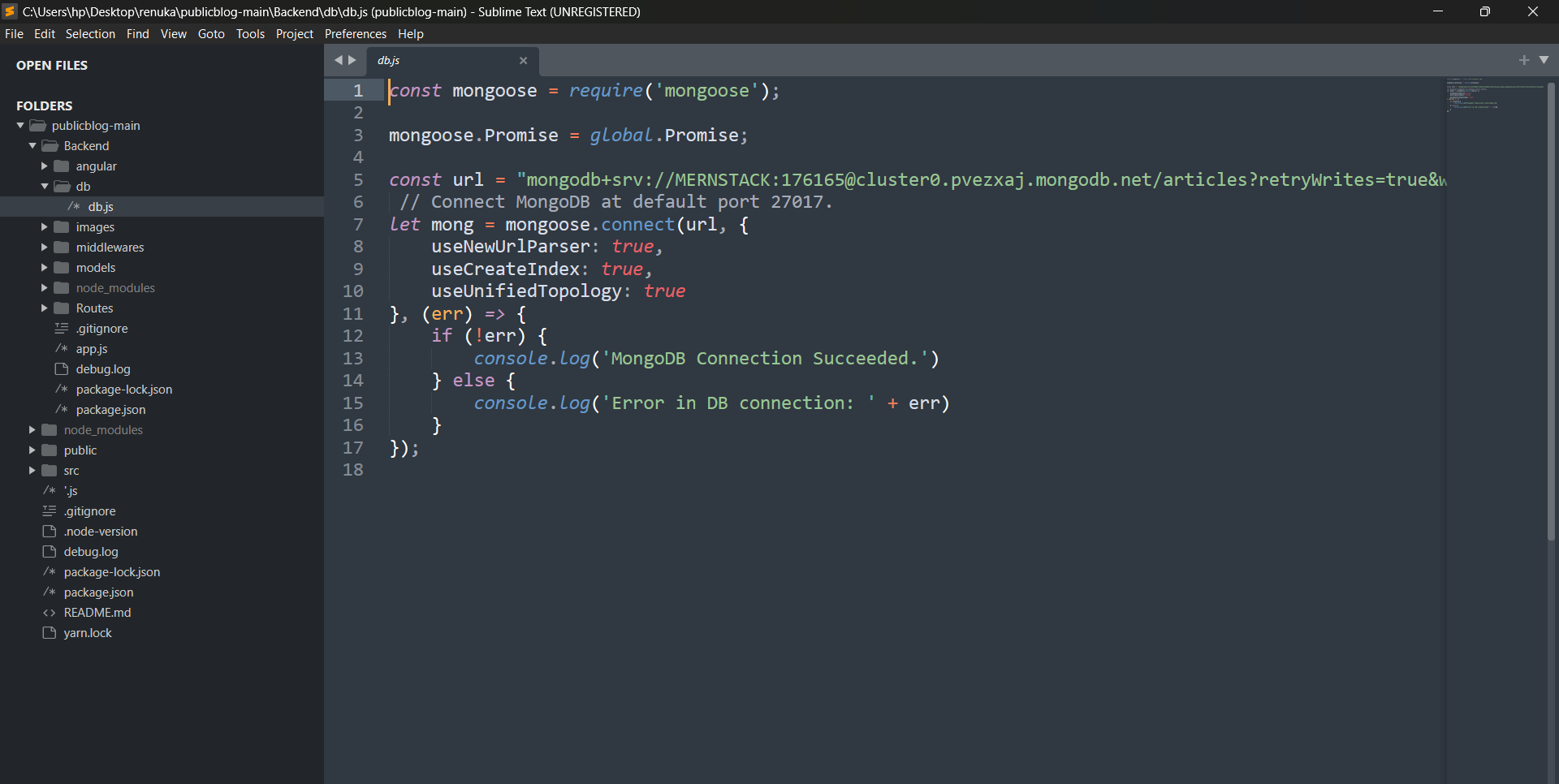
Features of the website:

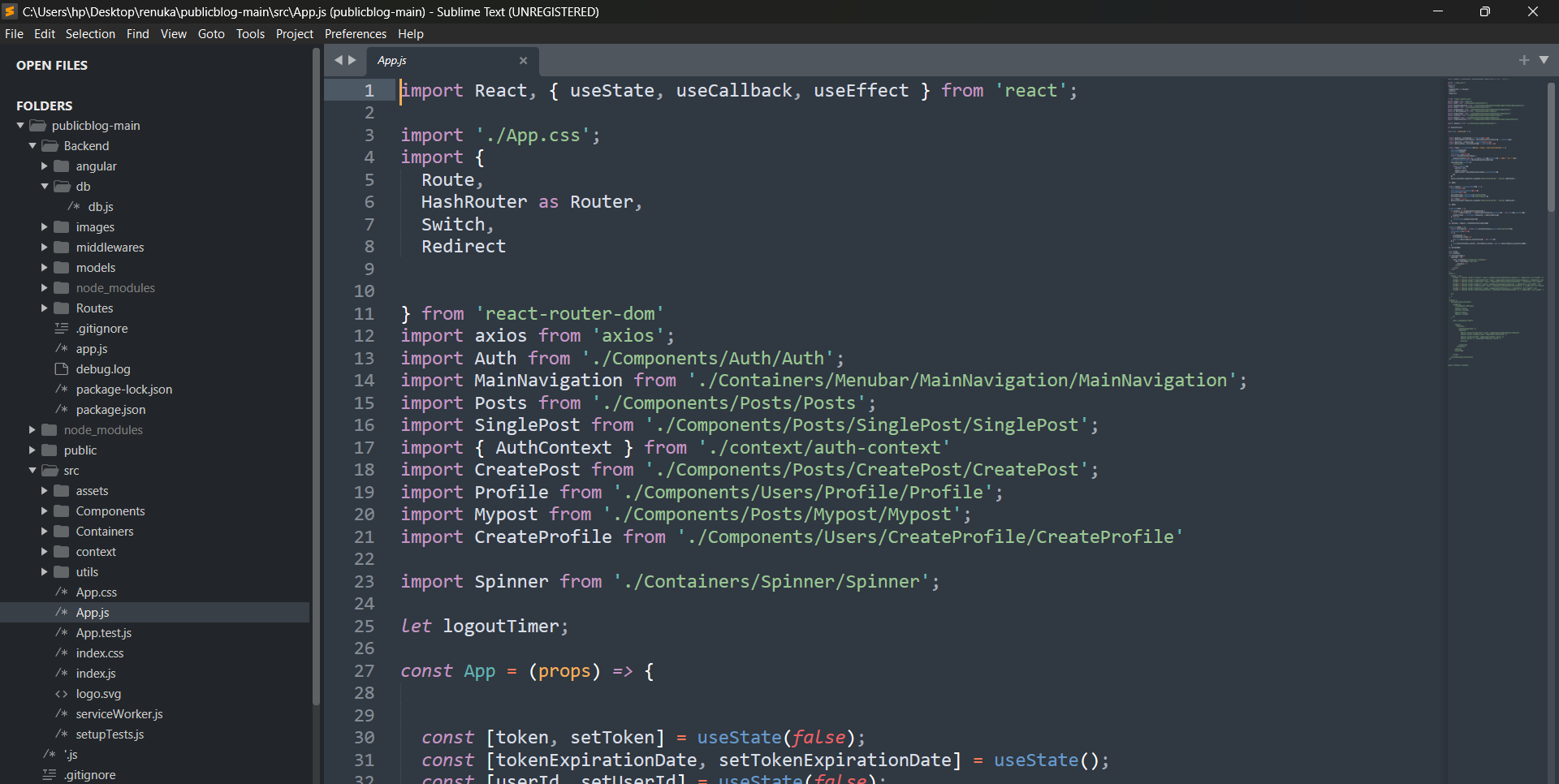
* Authentication
* Log in/Sign up
* Create or delete a post
* Update a post
* Security
* Structural design
* User friendly

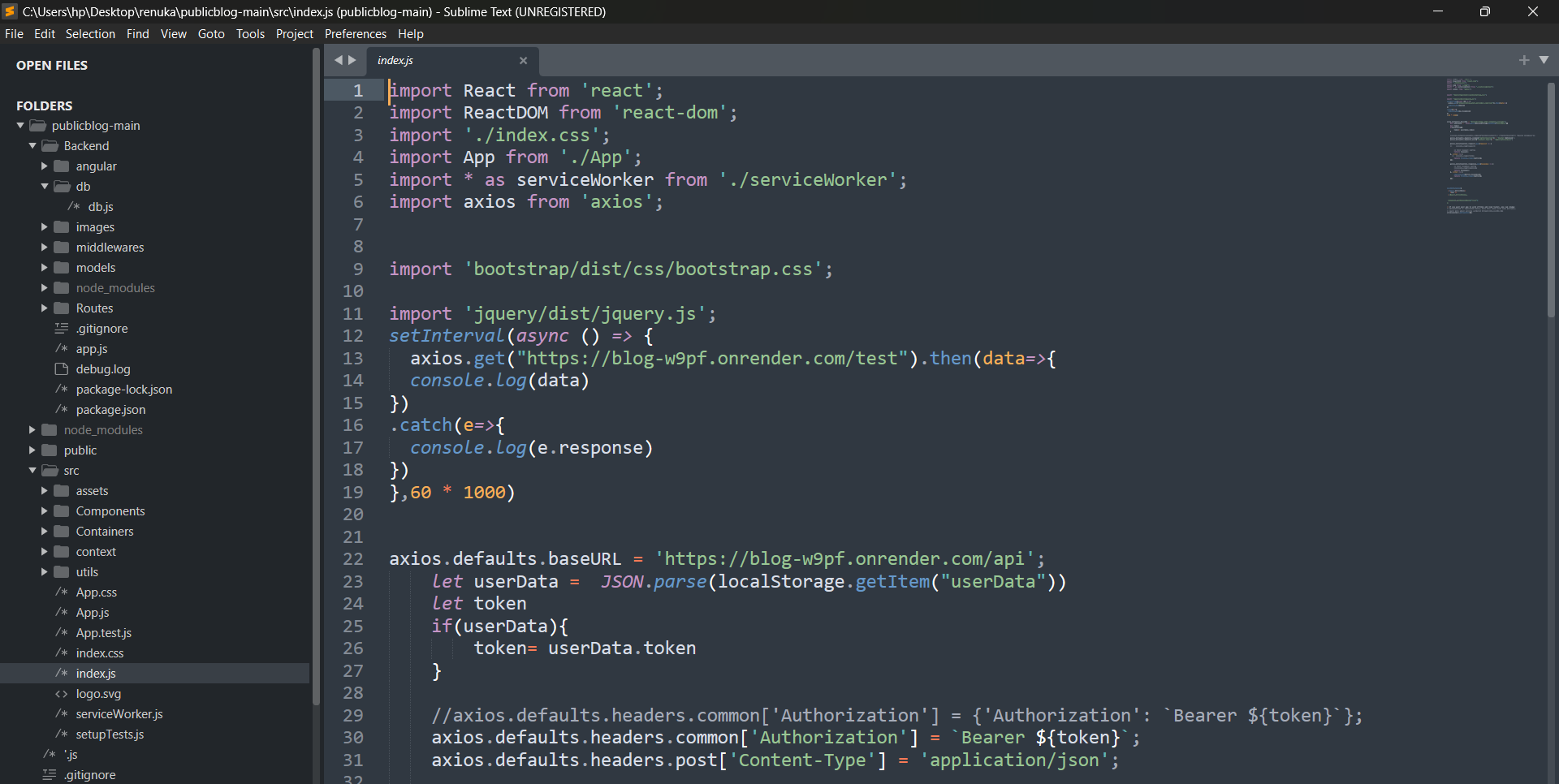
**SAMPLE CODE:**

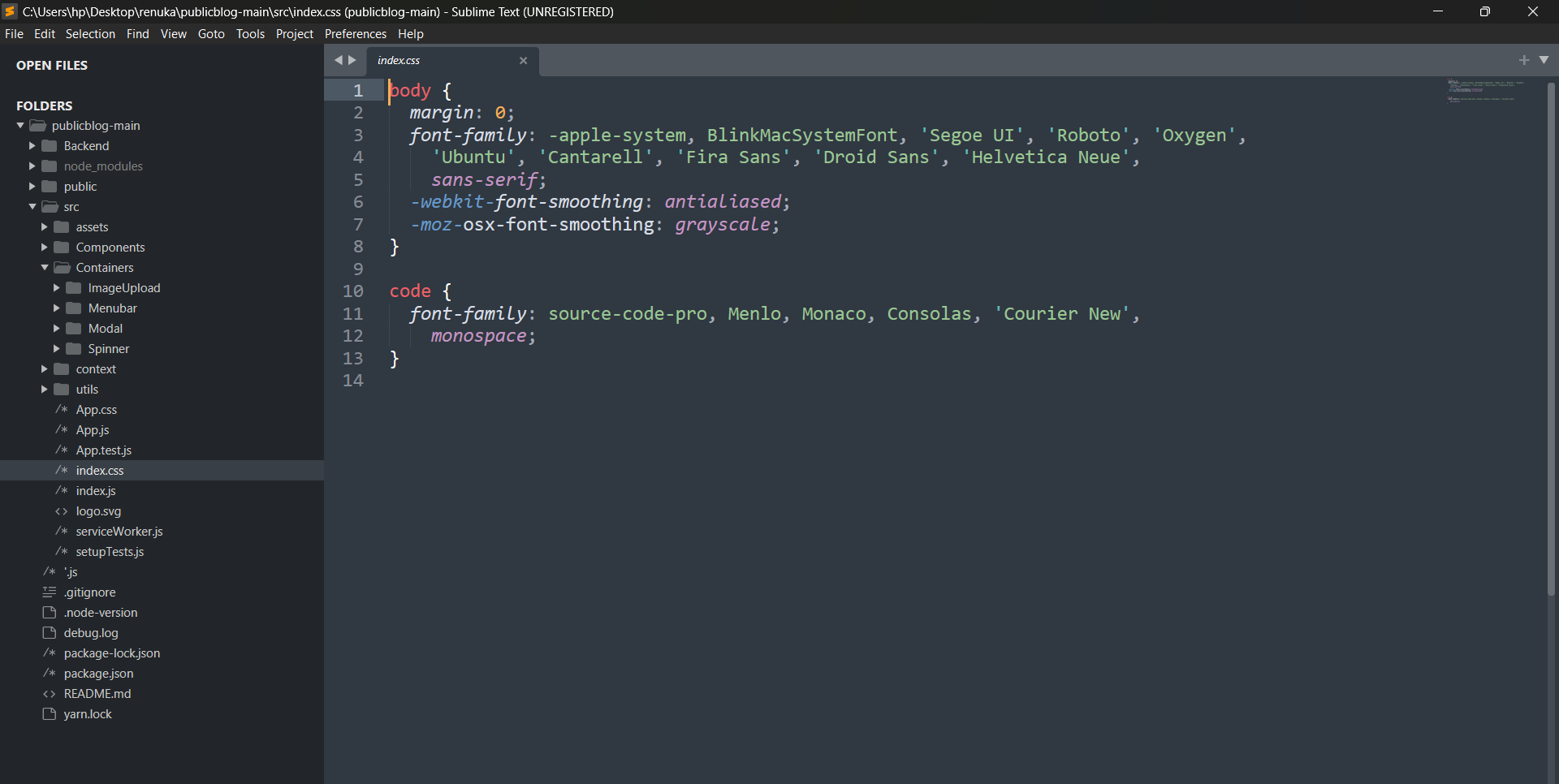
****

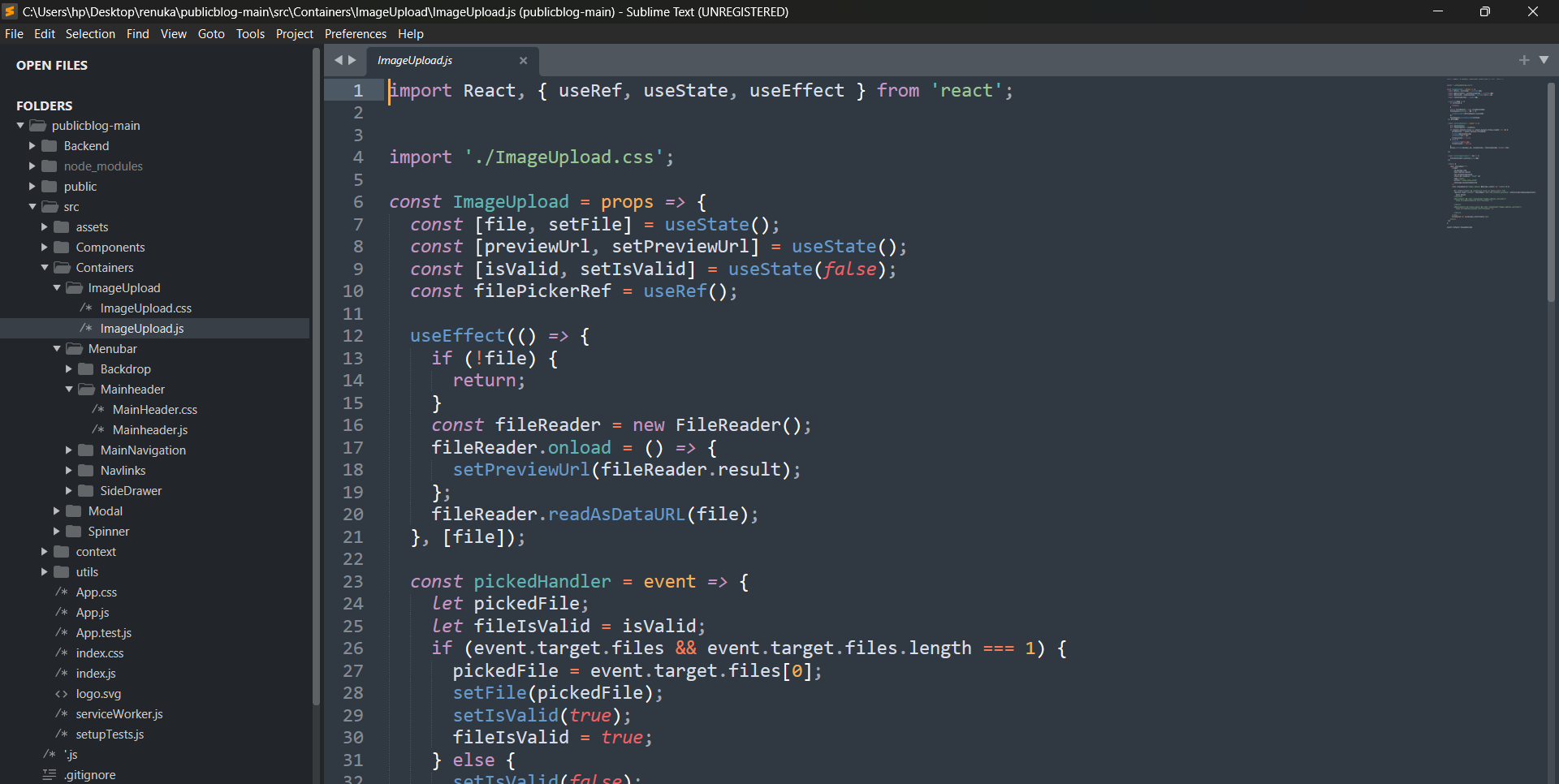
****

****

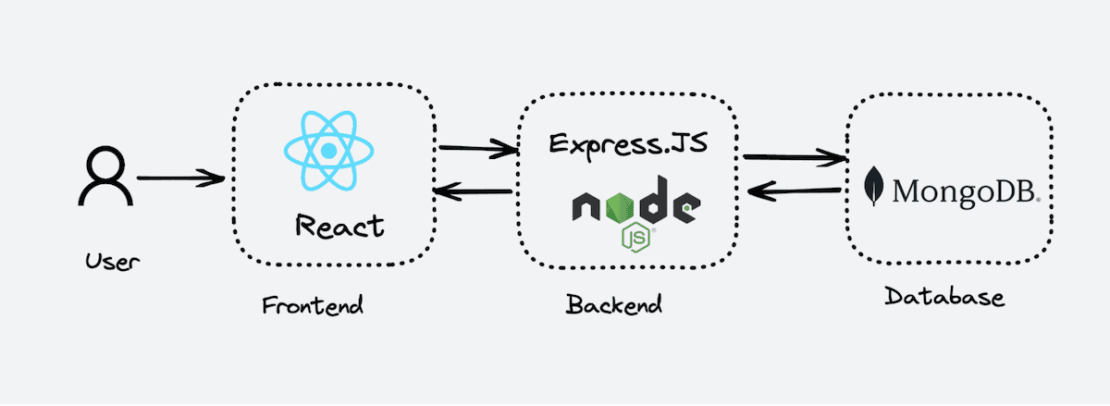
****

****

****

****

**Software Engineering tools used:**





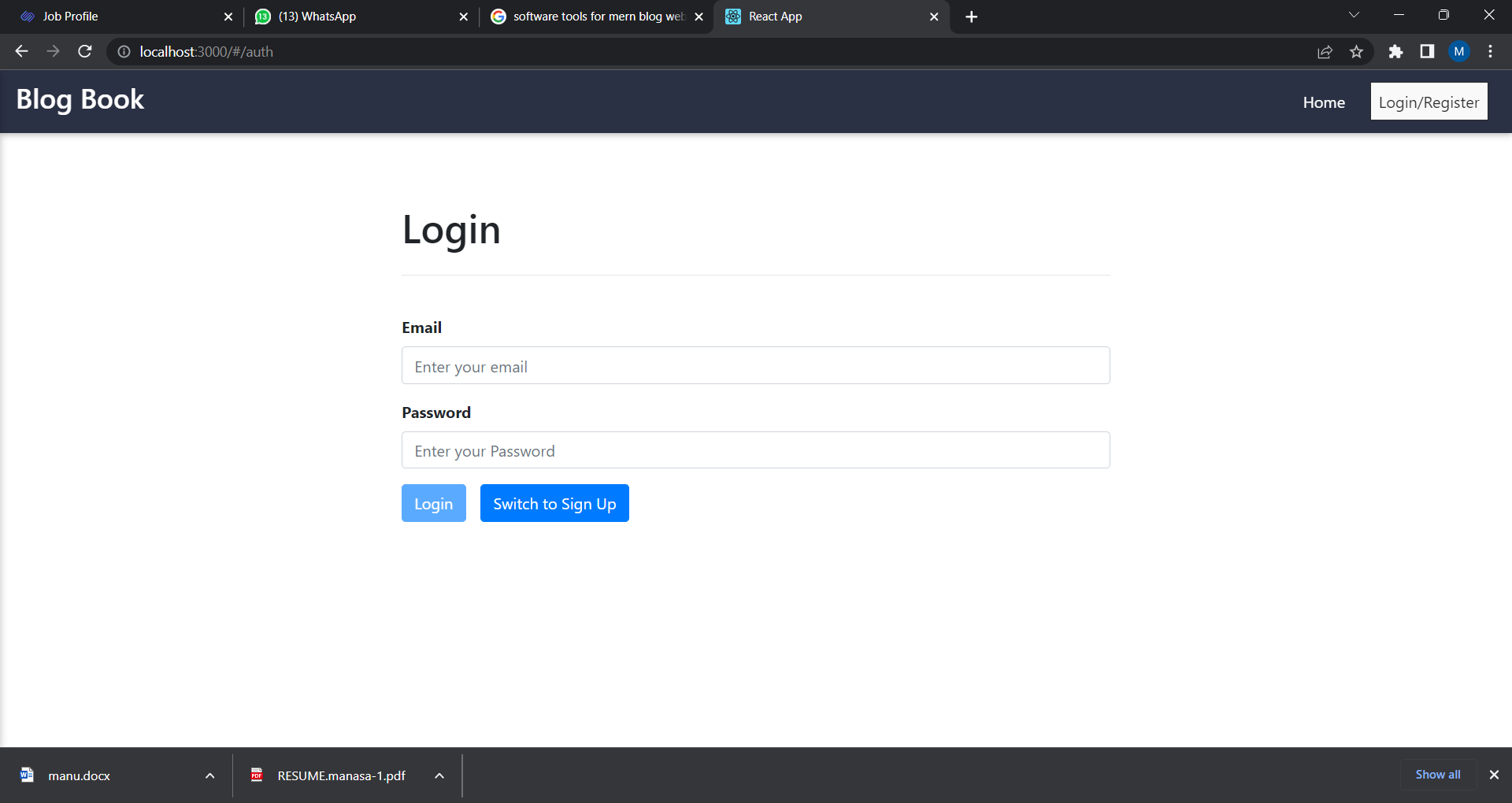
**Unit Testing:**

Tools used and its setup details:

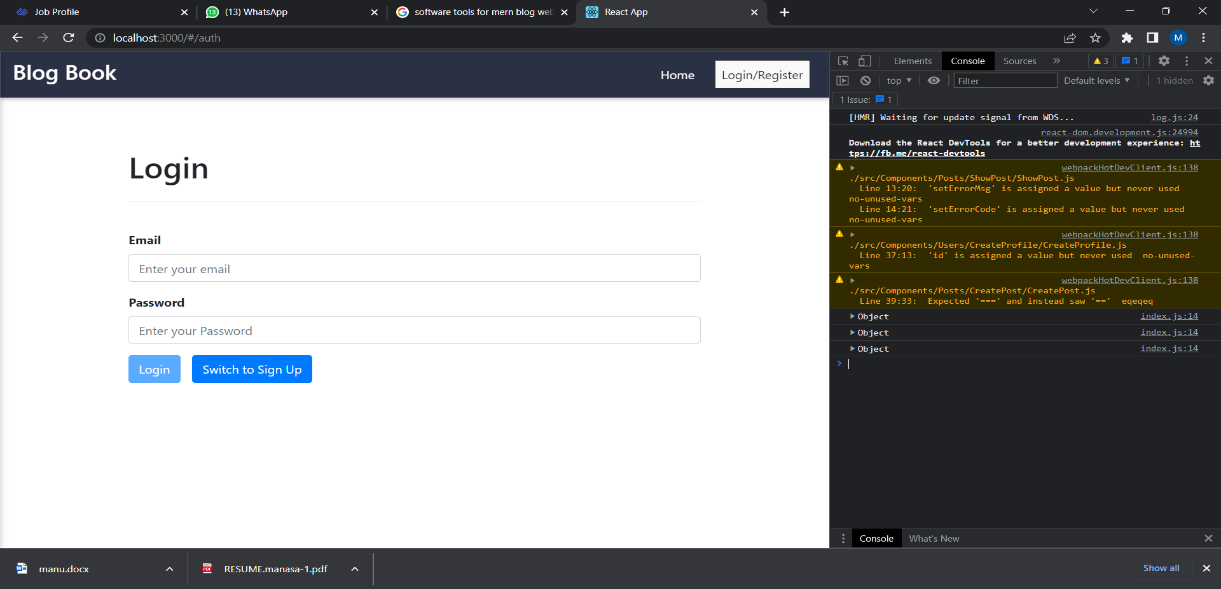
* axios: "^0.20.0"
* bootstrap: "^4.5.2"
* jquery: "^3.5.1"
* moment: "^2.29.0"
* react: "^16.13.1"
* react-dom: "^16.13.1"
* react-router-dom: "^5.2.0"
* react-scripts: "3.4.3"
* "bcrypt": "^5.0.0"
* body-parser: "^1.19.0"
* cors: "^2.8.5",
* express: "^4.17.1"
* jsonwebtoken: "^8.5.1"
* mongoose: "^5.10.4"
* multer: "^1.4.2"

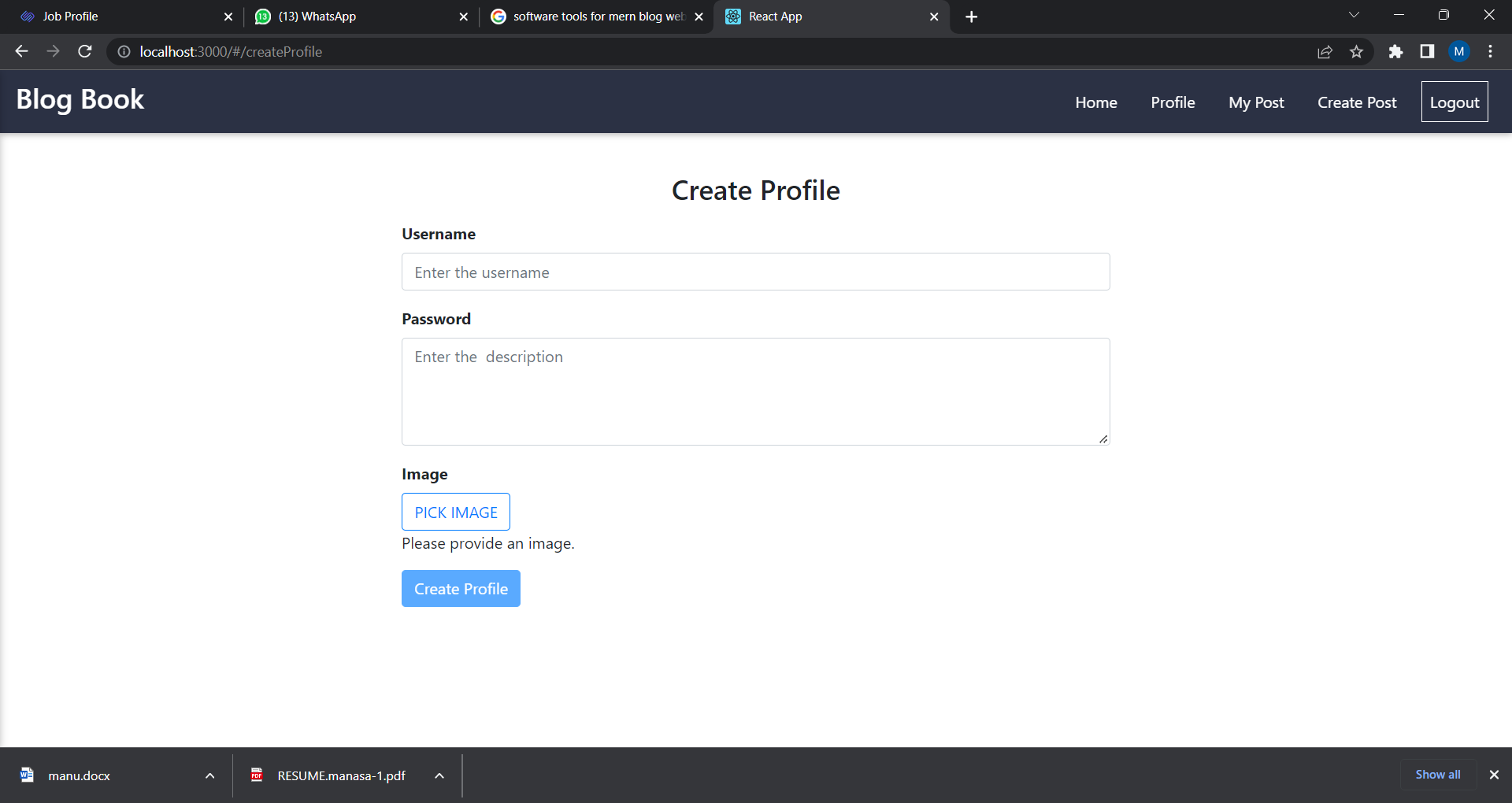
**UI Testing:**

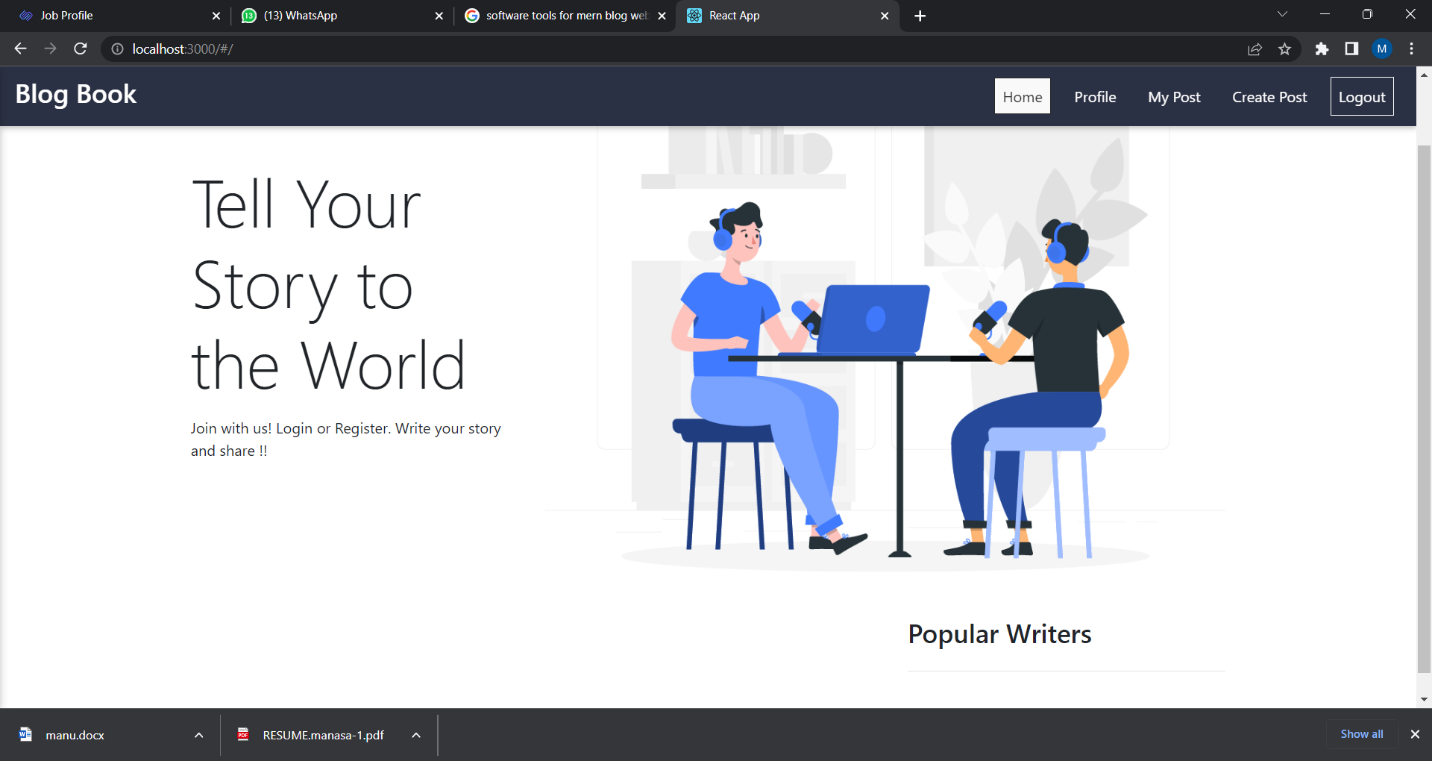
LOGIN PAGE:



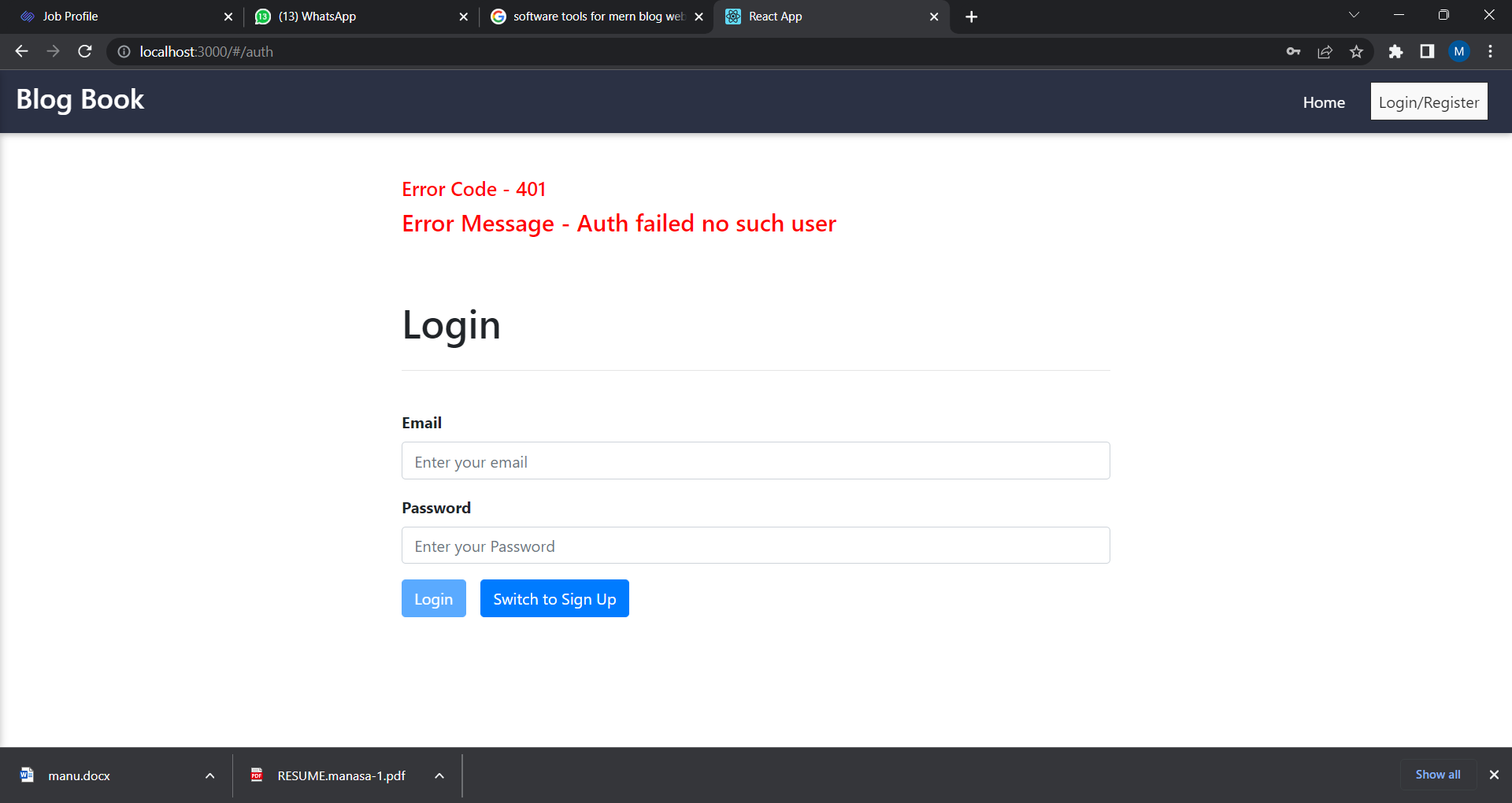
Testing Login Page:







Testing Incorrect Credentials:



import React, { useState, useCallback, useEffect } from 'react';

import './App.css';

import {

Route,

HashRouter as Router,

Switch,

Redirect

} from 'react-router-dom'

import axios from 'axios';

import Auth from './Components/Auth/Auth';

import MainNavigation from './Containers/Menubar/MainNavigation/MainNavigation';

import Posts from './Components/Posts/Posts';

import SinglePost from './Components/Posts/SinglePost/SinglePost';

import { AuthContext } from './context/auth-context'

import CreatePost from './Components/Posts/CreatePost/CreatePost';

import Profile from './Components/Users/Profile/Profile';

import Mypost from './Components/Posts/Mypost/Mypost';

import CreateProfile from './Components/Users/CreateProfile/CreateProfile'

import Spinner from './Containers/Spinner/Spinner';

let logoutTimer;

const App = (props) => {

const [token, setToken] = useState(false);

const [tokenExpirationDate, setTokenExpirationDate] = useState();

const [userId, setUserId] = useState(false);

const [isLoading, setIsloading] = useState(true)

const login = useCallback((uid, token, expirationDate) => {

setToken(token);

setUserId(uid);

setIsloading(false)

const tokenExpirationDate =

expirationDate || new Date(new Date().getTime() + 1000 \* 60 \* 60);

setTokenExpirationDate(tokenExpirationDate);

localStorage.setItem(

'userData',

JSON.stringify({

userId: uid,

token: token,

expiration: tokenExpirationDate.toISOString()

})

);

axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

}, []);

const logout = useCallback(() => {

setToken(null);

setTokenExpirationDate(null);

setUserId(null);

localStorage.removeItem('userData');

localStorage.removeItem('profileData');

let token = null

axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

}, []);

useEffect(() => {

if (token && tokenExpirationDate) {

const remainingTime = tokenExpirationDate.getTime() - new Date().getTime();

logoutTimer = setTimeout(logout, remainingTime);

} else {

clearTimeout(logoutTimer);

}

}, [token, logout, tokenExpirationDate]);

useEffect(() => {

const storedData = JSON.parse(localStorage.getItem('userData'));

setIsloading(false)

if (

storedData &&

storedData.token &&

new Date(storedData.expiration) > new Date()

) {

login(storedData.userId, storedData.token, new Date(storedData.expiration));

}

}, [login]);

let route

let loading

if (isLoading) {

loading = (<>

<div className="container loading">

<div className="mar-20">

<Spinner />

</div>

</div>

</>)

}

else {

route = (<>

{token ? <Route path="/create" exact component={CreatePost}></Route> : <Redirect to="/auth" />}

{token ? <Route path="/createProfile" exact component={CreateProfile}></Route> : <Redirect to="/auth" />}

{token ? <Route path="/edit/:id" exact component={CreatePost}></Route> : <Redirect to="/auth" />}

{token ? <Route path="/mypost" exact component={Mypost}></Route> : <Redirect to="/auth" />}

{token ? <Route path="/mypost/:id" exact component={SinglePost}></Route> : <Redirect to="/auth" />}

{token ? <Route path="/profile" exact component={Profile} /> : <Redirect to="/auth" />}

{token ? <Route path="/profile/edit/:id" component={CreateProfile} /> : <Redirect to="/auth" />}

</>

)

}

return (

<AuthContext.Provider

value={{

isLoggedIn: !!token,

token: token,

userId: userId,

login: login,

logout: logout

}}

>

<div className="App">

<main>

<Router>

<MainNavigation />

<Switch>

<Route path="/post/:id" exact component={SinglePost}></Route>

<Route path="/public/:id" component={Profile} />

<Route path="/auth" component={Auth} exact />

<Route path="/" component={Posts} exact />

{route}

</Switch>

</Router>

</main>

{loading}

</div>

</AuthContext.Provider>

);

}

export default (App);;

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import \* as serviceWorker from './serviceWorker';

import axios from 'axios';

import 'bootstrap/dist/css/bootstrap.css';

import 'jquery/dist/jquery.js';

setInterval(async () => {

axios.get("https://blog-w9pf.onrender.com/test").then(data=>{

console.log(data)

})

.catch(e=>{

console.log(e.response)

})

},60 \* 1000)

axios.defaults.baseURL = 'https://blog-w9pf.onrender.com/api';

let userData = JSON.parse(localStorage.getItem("userData"))

let token

if(userData){

token= userData.token

}

//axios.defaults.headers.common['Authorization'] = {'Authorization': `Bearer ${token}`};

axios.defaults.headers.common['Authorization'] = `Bearer ${token}`;

axios.defaults.headers.post['Content-Type'] = 'application/json';

axios.interceptors.request.use(request => {

// console.log(request)

// Edit request config

return request;

}, error => {

// console.log(error);

return Promise.reject(error);

});

axios.interceptors.response.use(response => {

// Edit response config

//console.log(response);

return response;

}, error => {

console.log(error.response);

return Promise.reject(error);

});

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

FOR IMAGES UPLOAD:

import React, { useRef, useState, useEffect } from 'react';

import './ImageUpload.css';

const ImageUpload = props => {

const [file, setFile] = useState();

const [previewUrl, setPreviewUrl] = useState();

const [isValid, setIsValid] = useState(false);

const filePickerRef = useRef();

useEffect(() => {

if (!file) {

return;

}

const fileReader = new FileReader();

fileReader.onload = () => {

setPreviewUrl(fileReader.result);

};

fileReader.readAsDataURL(file);

}, [file]);

const pickedHandler = event => {

let pickedFile;

let fileIsValid = isValid;

if (event.target.files && event.target.files.length === 1) {

pickedFile = event.target.files[0];

setFile(pickedFile);

setIsValid(true);

fileIsValid = true;

} else {

setIsValid(false);

fileIsValid = false;

}

props.onInput(props.id, pickedFile, fileIsValid); return file

};

const pickImageHandler = () => {

filePickerRef.current.click();

};

return (

<div className="">

<input

id={props.id}

name={props.name}

ref={filePickerRef}

style={{ display: 'none' }}

type="file"

accept=".jpg,.png,.jpeg"

onChange={pickedHandler}

/>

<div className={`image-upload ${props.center && 'center'}`}>

{/\* {!previewUrl && <p>Please pick an image.</p>} \*/}

<button type="button" className="btn btn-outline-primary" onClick={pickImageHandler}>

PICK IMAGE

</button>

{previewUrl && <div className="image-upload\_\_preview">

<img src={previewUrl} alt="Preview" />

</div>}

{!previewUrl && props.value && <div className="image-upload\_\_preview">

<img src={props.value} alt="Preview" />

</div>}

</div>

{!isValid && <p>{props.errorText}</p>}

</div>

);

};

export default ImageUpload;

AUTHENTICATION:

import Axios from 'axios';

import React, { Component } from 'react'

import { withRouter } from 'react-router-dom'

import validateForm from '../../utils/validateform'

import validEmailRegex from '../../utils/emailRegex'

import './Auth.css'

import { AuthContext } from '../../context/auth-context'

import Spinner from '../../Containers/Spinner/Spinner';

export class Auth extends Component {

static contextType = AuthContext

constructor(props) {

super(props)

this.state = {

user: {

email: '',

password: ''

},

error: {

message: '',

code: ''

},

isloading: false,

isLoginMode: true,

errors: {

email: '',

password: ''

}

}

}

mySubmitHandler = (event) => {

this.setState(pre => ({

isloading: true

}))

const auth = this.context

event.preventDefault();

if (validateForm(this.state.errors)) {

} else {

}

if (this.state.isLoginMode) {

Axios.post('/user/login', this.state.user)

.then(response => {

this.setState(pre => ({

isloading: false

}))

this.props.history.push('/')

auth.login(response.data.userId, response.data.token);

return Axios.get('/profile/viewprofile')

}).then(data => {

let profile = data.data.profile.username

localStorage.setItem(

'profileData',

JSON.stringify({

"username": profile

}))

}).catch(e => {

this.setState({

isloading: false,

error: {

...this.state.error, message: e.response.data.message,

code: e.response.status

}

});

})

}

else {

this.setState(pre => ({

isloading: true

}))

Axios.post('/user/signup', this.state.user).then(response => {

this.setState(pre => ({

isloading: false

}))

})

.catch(e => {

this.setState({ error: true });

})

}

this.setState({

user: { ...this.state.user, email: '', password: '' }

});

}

myChangeHandler = (event) => {

let nam = event.target.name;

let val = event.target.value;

let errors = this.state.errors;

const { name, value } = event.target;

switch (name) {

case 'email':

if (value.length === 0) {

errors.email =

value.length < 5

? 'Email is Required!'

: '';

break;

}

if (value.length > 0) {

errors.email =

validEmailRegex.test(value)

? ''

: 'Email is not valid!';

break;

}

break;

case 'password':

if (value.length > 0) {

errors.password =

value.length < 6

? 'Password must be 6 characters long!'

: '';

}

if (value.length === 0) {

errors.password =

value.length === 0

? 'Password is required!'

: '';

}

break;

default:

break;

}

this.setState({ errors, user: { ...this.state.user, [nam]: val } }, () => {

});

}

switchLoginhandler = () => {

this.setState(pre => ({

isLoginMode: !pre.isLoginMode

}))

}

render() {

let isLoading

let iserror

if (this.state.isloading) {

isLoading = (

<>

<div className="container loading">

<div className="mar-20">

<Spinner />

</div>

</div>

</>

)

}

if (this.state.error.code) {

iserror = (

<>

<div className="container error container-short">

<div className="mar-20">

<h5>Error Code - {this.state.error.code}</h5>

<h4>Error Message - {this.state.error.message}</h4>

</div>

</div>

</>

)

}

return (<>

{isLoading}

{iserror}

<div className="container container-short py-5">

<h1 className="pt-2 py-2">{this.state.isLoginMode ? 'Login ' : 'Sign Up'}</h1>

<hr></hr>

<form onSubmit={this.mySubmitHandler} className="pt-4">

<div className="form-group">

<label htmlFor="email">Email </label>

<input

type='email'

name='email'

value={this.state.user.email}

className={"form-control " + (this.state.errors.email ? 'is-invalid' : '')}

placeholder="Enter your email"

required

onChange={this.myChangeHandler}

/>

{this.state.errors.email.length > 0 &&

<div className="mt-1"><span className='error text-danger'>{this.state.errors.email}</span></div>}

</div>

<div className="form-group">

<label htmlFor="password">Password </label>

<input

type='password'

name='password'

value={this.state.user.password}

className={"form-control " + (this.state.errors.password ? 'is-invalid' : '')}

placeholder="Enter your Password"

required="required"

data-error="Please enter your full name."

onChange={this.myChangeHandler}

/>

{this.state.errors.password.length > 0 &&

<div className="mt-1"> <span className='error text-danger'>{this.state.errors.password}</span></div>}

</div>

<div className="form-group">

<button style={{ marginRight: '15px' }}

type='submit'

className="btn btn-primary"

disabled={this.state.user.email && this.state.user.password

&& (validateForm(this.state.errors)) ? '' : 'disabled'}

>

{this.state.isLoginMode ? 'Login' : 'Sign Up'}

</button>

<button

type='button'

className="btn btn-primary"

onClick={this.switchLoginhandler}

>Switch to {this.state.isLoginMode ? 'Sign Up' : 'Login'} </button>

</div>

</form>

</div>

</>

)

}

}

export default withRouter(Auth)

CHECKING AUTHENTICATION:

const jwt = require("jsonwebtoken");

module.exports = (req, res, next) => {

try {

const token = req.headers.authorization.split(" ")[1];

const decodedToken = jwt.verify(token, "secret\_this\_should\_be\_longer");

req.userData = { email: decodedToken.email, userId: decodedToken.userId };

next();

} catch (error) {

res.status(401).json({ message: "Auth failed!" });

}

};

ROUTES:

const express = require('express')

const Post = require('../models/post')

const router = new express.Router()

const multer = require("multer");

const checkAuth = require("../middlewares/check-auth");

const MIME\_TYPE\_MAP = {

"image/png": "png",

"image/jpeg": "jpg",

"image/jpg": "jpg",

"image/gif": "gif"

};

const storage = multer.diskStorage({

destination: (req, file, cb) => {

console.log(req,file)

const isValid = MIME\_TYPE\_MAP[file.mimetype];

let error = new Error("Invalid mime type");

if (isValid) {

error = null;

}

cb(error, "images");

},

filename: (req, file, cb) => {

const name = file.originalname

.toLowerCase()

.split(" ")

.join("-");

console.log(name)

const ext = MIME\_TYPE\_MAP[file.mimetype];

cb(null, name + "-" + Date.now() + "." + ext);

}

});

router.post("",

checkAuth,

multer({ storage: storage }).single("image"),

(req, res, next) => {

console.log(req.body)

const url = req.protocol + "://" + req.get("host")

console.log(url)

const post = new Post({

title: req.body.title,

content: req.body.content,

imagePath: url + "/images/" + req.file.filename,

creator: req.userData.userId,

postDate: req.body.postDate,

})

console.log(post)

post.save().

then(post => {

if(post){

res.status(201).json({

message: "Post added successfully",

post: {

...post,

id: post.\_id

}

})

}

if(!post){

res.status(404).json({

message: "Error Adding Post",

})

}

})

.catch(e => {

console.log(e)

res.status(501).json({ message: "Error Adding Post"+e });

})

})

router.put(

"/:id",

checkAuth,

multer({ storage: storage }).single("image"),

(req, res, next) => {

let imagePath = req.body.imagePath;

if (req.file) {

const url = req.protocol + "://" + req.get("host");

imagePath = url + "/images/" + req.file.filename

}

console.log("90",req.body)

const post = new Post({

\_id: req.body.id,

title: req.body.title,

content: req.body.content,

imagePath: imagePath,

creator: req.userData.userId

});

console.log("98---------------------",post);

Post.updateOne(

{ \_id: req.params.id, creator: req.userData.userId },

post

).then(result => {

if(result){

res.status(200).json({ message: "Update successful!" });

}

else {

res.status(500).json({ message: "Error Upating Post" });

}

});

}

);

router.get("/mypost",

checkAuth,

(req, res, next) => {

Post.find({creator: req.userData.userId}).then(post => {

if (post) {

res.status(200).json({

message: "Posts fetched successfully!",

posts: post

});

} else {

res.status(404).json({ message: "Post not found!" });

}

})

.catch(e=>{

console.log(e)

});

});

router.get("", (req, res, next) => {

Post.find().then(documents => {

if(documents){

res.status(200).json({

message: "Posts fetched successfully!",

posts: documents

});

}

else{

res.status(404).json({ message: "Post not found!" });

}

});

});

router.get("/:id", (req, res, next) => {

Post.findById(req.params.id).then(post => {

if (post) {

res.status(200).json(post);

} else {

res.status(404).json({ message: "Post not found!" });

}

});

});

router.delete("/:id", checkAuth, (req, res, next) => {

Post.deleteOne({ \_id: req.params.id, creator: req.userData.userId }).then(

result => {

console.log(result);

if (result.n > 0) {

res.status(200).json({ message: "Deletion successful!" });

} else {

return res.status(401).json({ message: "Not authorized!!" });

}

}

);

});

module.exports = router

**Conclusion**

With the MERN stack you can ideally build any web application you want by learning just one language, Javascript. With increased popularity of NoSQL databases, MongoDB is a go to database because of its scalability and flexible document schemas.

Though, the MERN stack is ideally suited for more JSON heavy, cloud native and dynamic web applications. One can build simple applications like the todo list, task manager to more complex ones like e-commerce sites and social media sites. MERN stack has growing popularity and many advantages with backing from a community of developers. If one aspires to be a full stack developer, he/she should definitely try out the MERN stack!

Finally, we are done for this MERN Stack Tutorial using React and Bootstrap 4. I have tried to highlight every essential topic in this tutorial. However, if you have skipped anything you can check out my [Github Repo](https://github.com/mehulk05/Blogapp-using-MERN" \t "_blank).