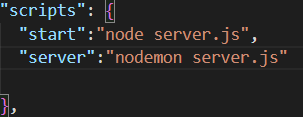
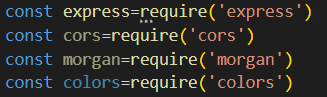
**Blog App Document**

# Create server

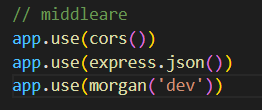
* First we have to create server by using command **npx init** , it will create **package.json** file into your directory which consist of details information about your project.
* Install some packages into your projects
* Make changes into your script to start server through nodemon
* Create server.js file into your project directory
* Import all packages into server.js file



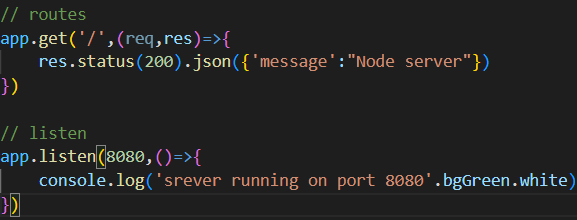
* Create rest object of app to add all functionality of express



* Create middleware for project

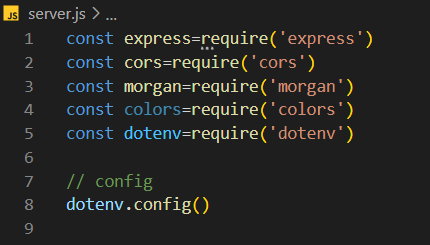


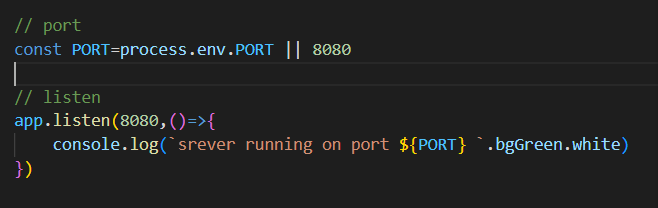
* Create routes into server.js file and also create listener by giving particular port



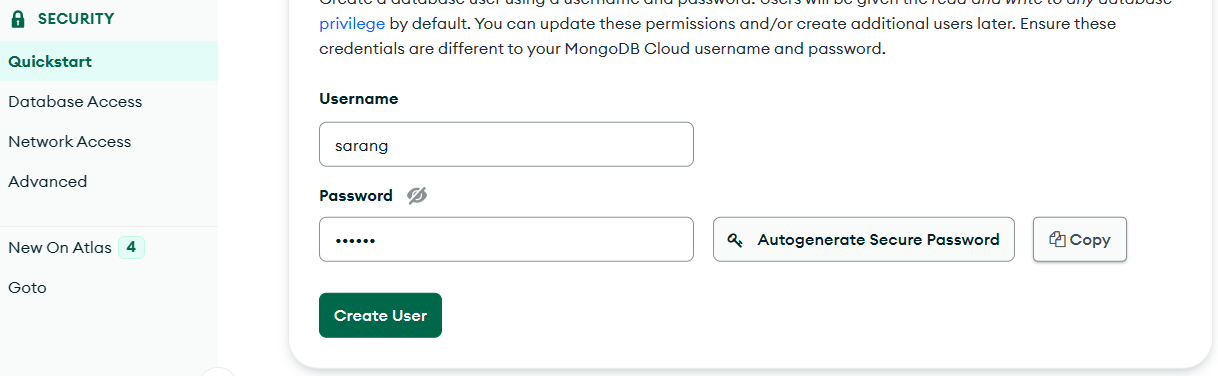
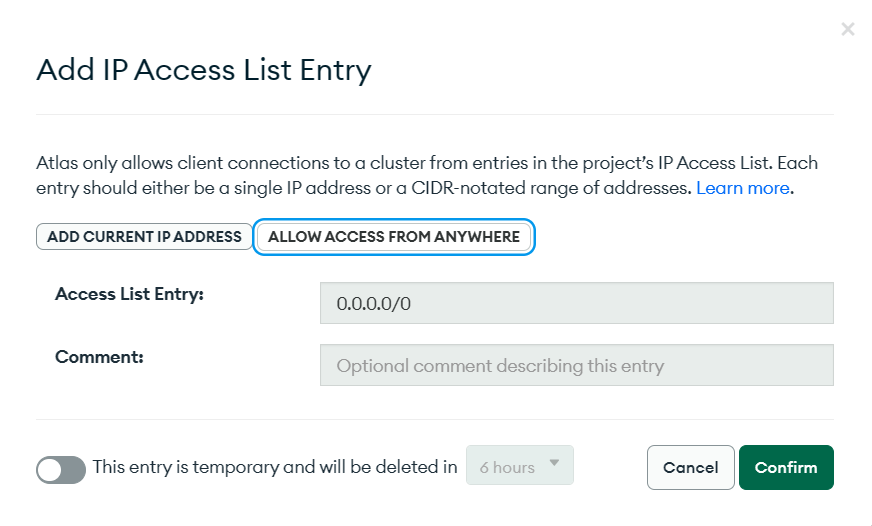
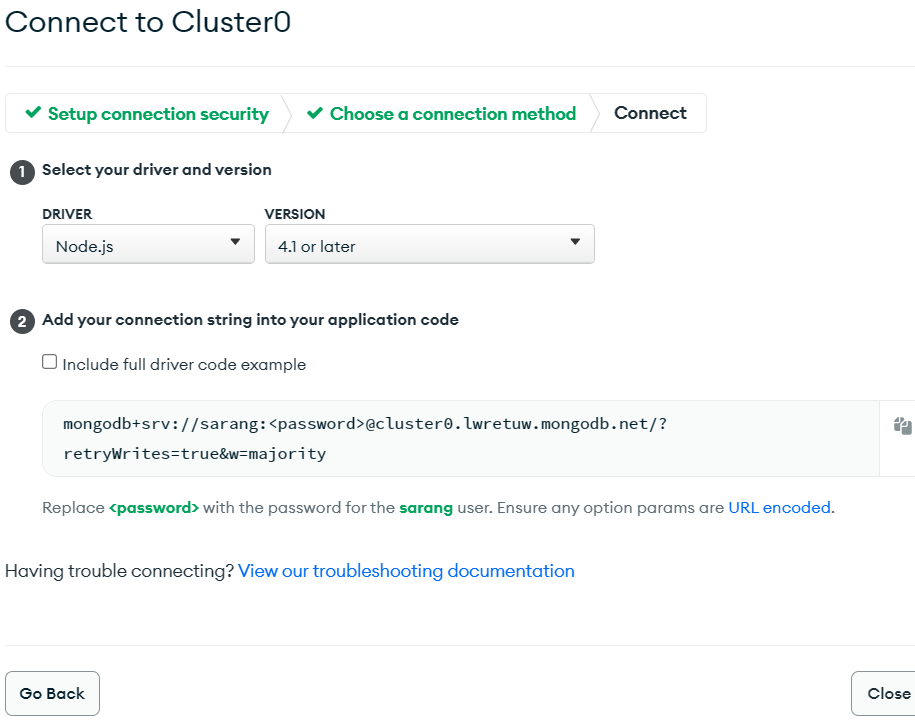
* Create dotenv file into root folder to hide your secrete credentials from to get public, so add tour port number init then import and configure dotenv into server.js file. Create seprate port onject and import port number from .env file



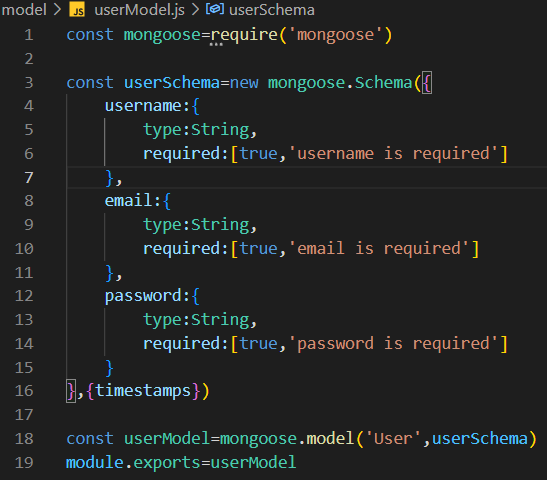
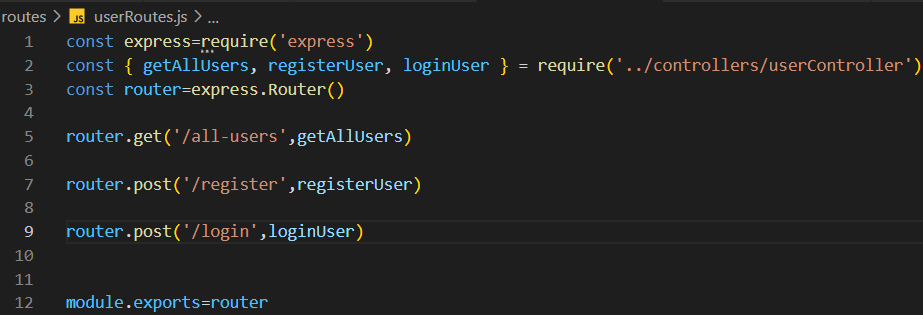




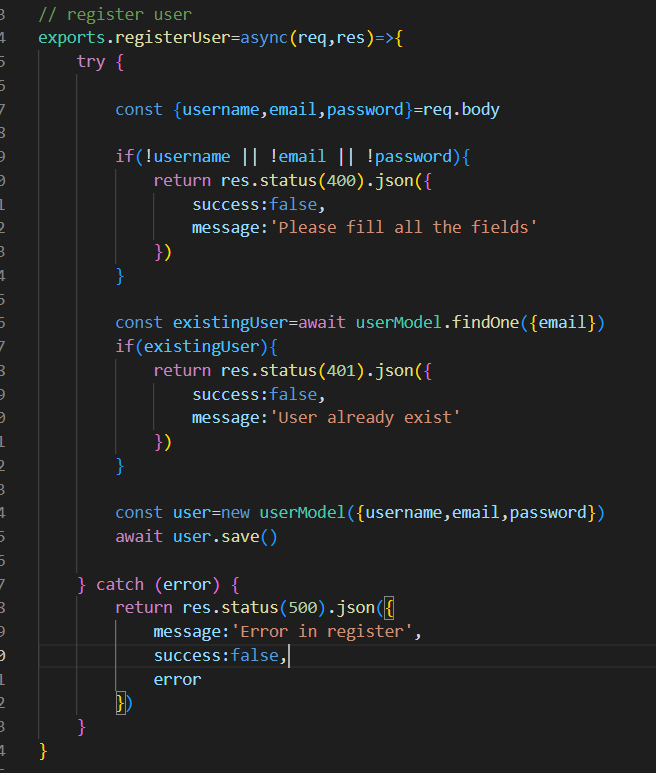
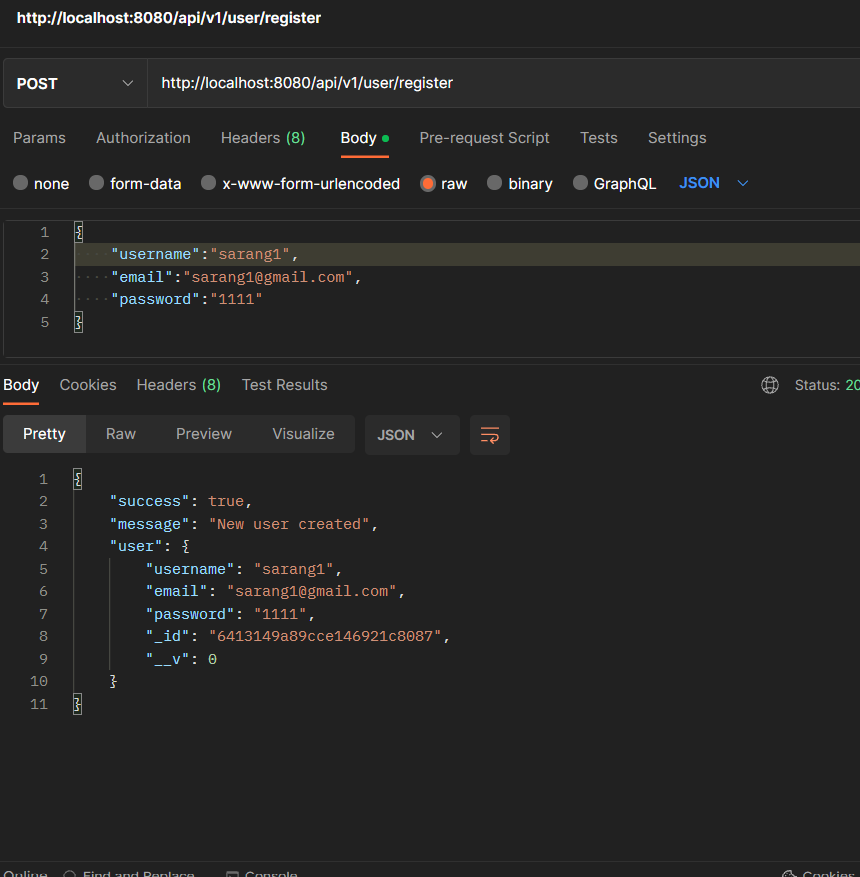
# Connect with mongodb cloud database

* Go to mongodb atlas website login there and create project after that create free cluster.
* Then in security quickstart add username and password
* Go to network add ip access ,click allow from anywhere and conform.
* Got to database click on connect option go to connect application copy mongo url and paste url into . env file 
* Add your actual password to ther url into .env file
* Create config folder into root and create db.js file
* Inside it we create connectDB function in which we import mongoose
* Using mongoose.connect() and passing url into it we connect wit DB
* At the end export function 
* Then import that function into server.js file 

# User model,routes and controller created

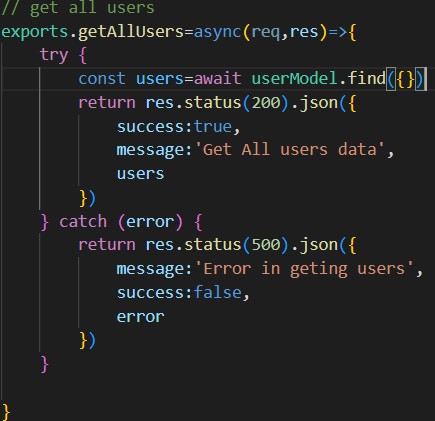
* Create model folder in root and create userModel.js file init
* Import mongoose and create userSchema by using mongoose.Schema({})
* Add all your field value insite schema and their type
* At the end use mongoose.model() method and pass field name and user schema init assign it to variable called uselModel
* Export userModel
* 
* Now create controller,routes folder into your root and also create userController.js file in controller and userRotes.js in routes
* Create multiple user routes by importing express then use express.routes() method and store into vasiavle route and export that variable
* Create multiple route using routes.get(),post(),put() and pass your endpoint and controller functions 
* Similirly create controllers funcitns in userController.js file and export it 

# Register controller API

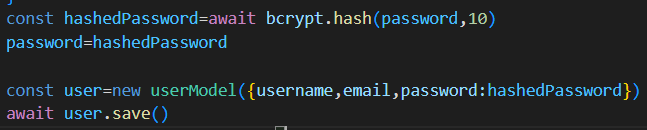
* First we make our callback fuction async and pass req,res to parameter
* We use try/catch block to handle errors
* In catch we send res status 500 and object which consist of error,message,and success status false.
* In try block firt we destrucy data which Is ckmming from user using req.body
* Then we validate weather there is data available or not if not thn we will send error message and ststus code 400
* Now we will check weather user already exist by importing userModel and check userModel.findone({}) by passing email weather user exist
* If user exist then we will send json res message and tell him user email already registered.
* If user not exist then we will save user by passing creadiantial inside userModel bu storing into variable and then use save() method. 
* You can send success res and status code 201
* Now you can check your api through postman weather user able to create account or not
* 

# Get all user callback apis

* Make callback func async/await and create try /catch block to handle error and send appropriate res from catch block
* userModel.find({}) method to get all the users store into variable and send res status and msg



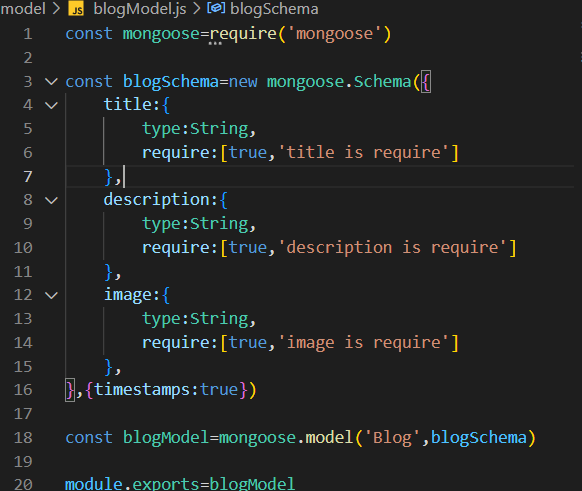
# Password hashing

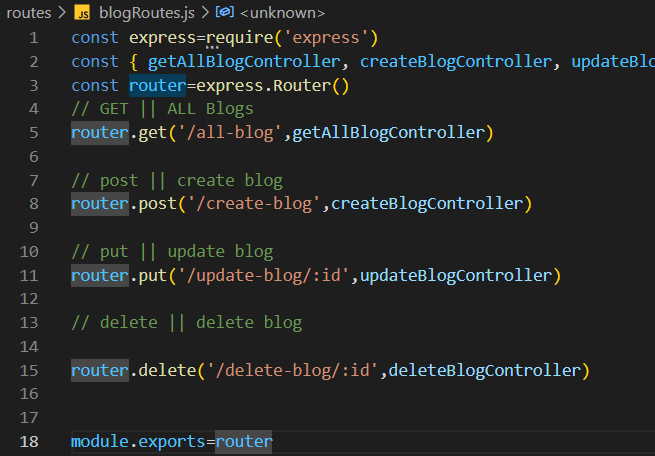
* first install package bcrypt 
* before saving users details in db just bcrypt password by using bcrypt.hash() inside it pass your password and salt value 

# User Login callback APIs

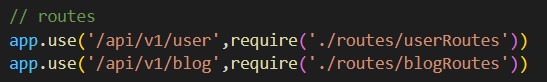
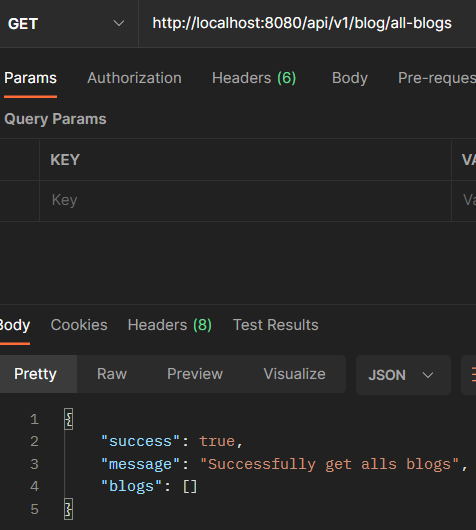
* First we make our callback fuction async and pass req,res to parameter
* We use try/catch block to handle errors
* In catch we send res status 500 and object which consist of error,message,and success status false.
* In try block first destructure email and password through req.body
* After that validate if email or password is there or not,if not send error
* Then validate if user is registerd or not if not registerd then send error
* Now compare password coming from body with user.password , if it is not match send error msg
* At the end send appropriate res of successfully login 

# Blog model,routes ,controller apis

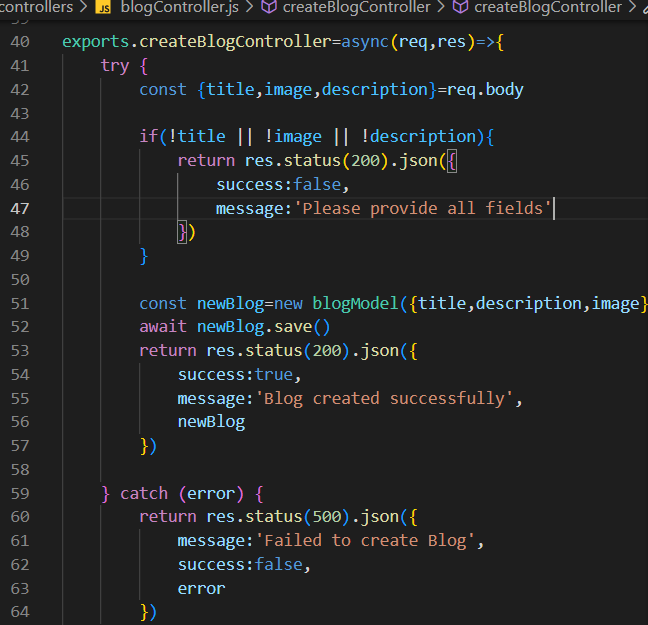
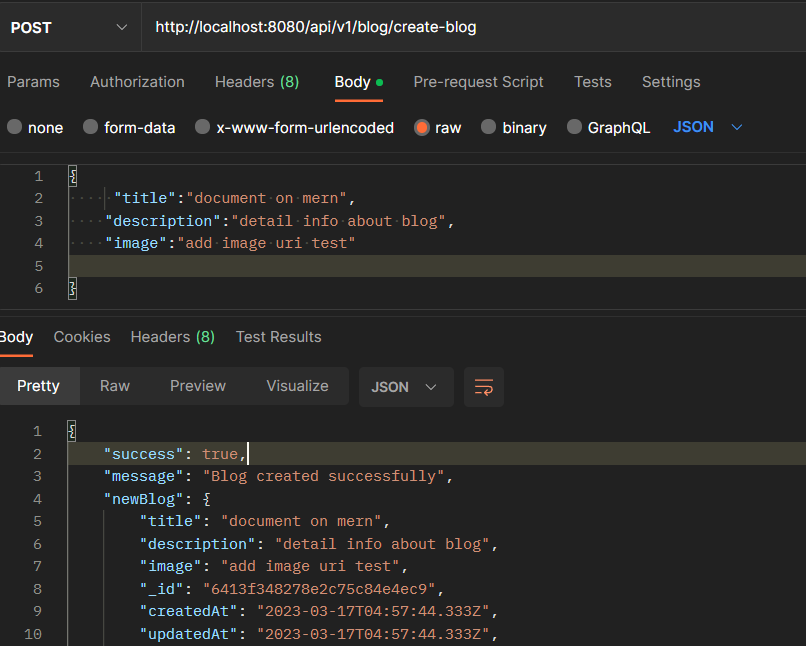
* Now create blog schema like user schema 
* Now create routes and controllers like user
* For blog we will create crud pattern routes

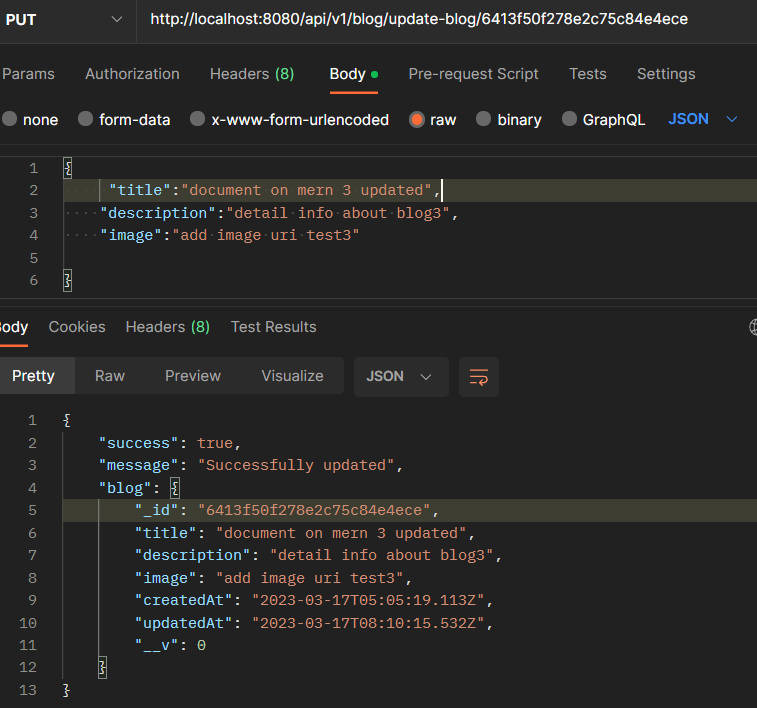
# Get all blogs controller apis

* Make it async /await and use try/catch blog and write appropriate res in catch blog
* In try blog get all blogs from blogModel and store into variable
* Validate if there is not blog send appropriate res with json
* Otherwise send success res 
* Add blog endpoint in server.js file 
* At the end check with postman weather your api working or not 

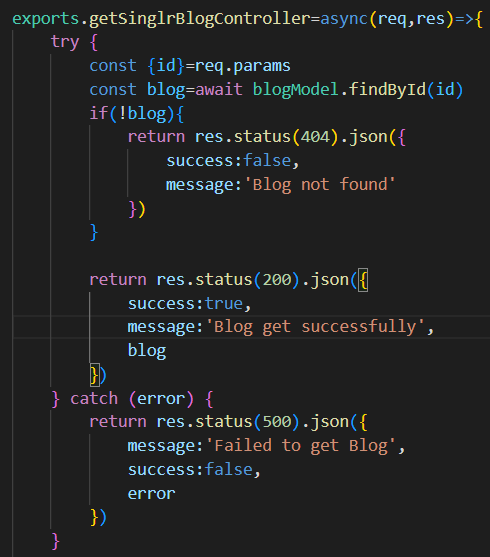
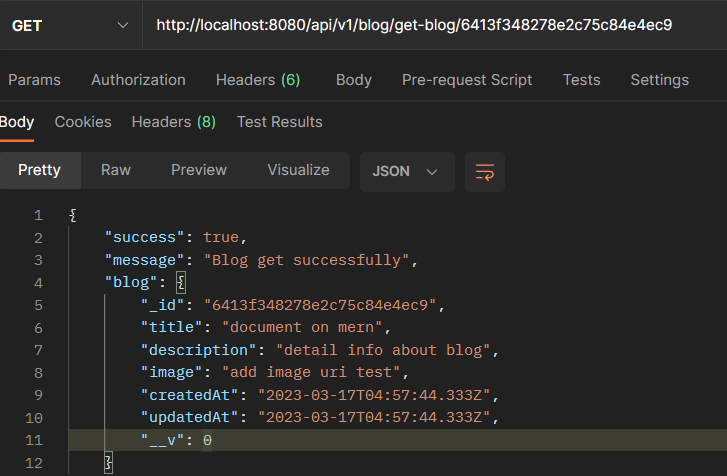
# Create new blog controller api’s

* Make it sync and use try/catch with res
* In try blog destructure req.body
* Now we validate if we get data or not,if not we will send error
* Otherwise all data pass into blogModel and save it and send success res 
* Now test api in postman 

# Update blog controller api

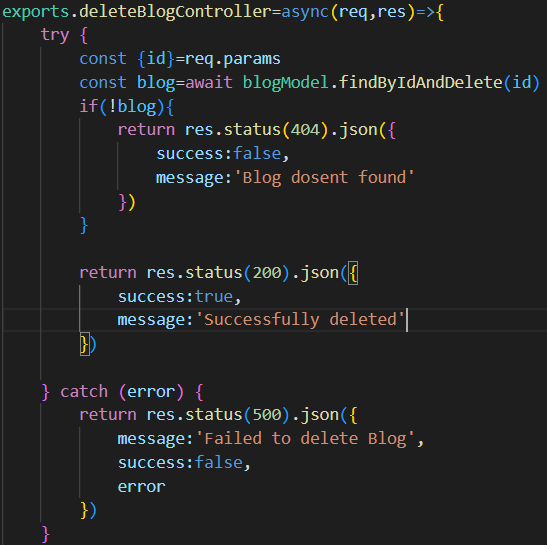
* For particular id we have to update blog
* First destructure id by using req.params method
* As well as destructure title description and image
* Now we will use blogModelfindByidAndUpdate() method inside it we will pass id ,new data,new object and send appropriate success res 
* Check in postman weather api working or not 

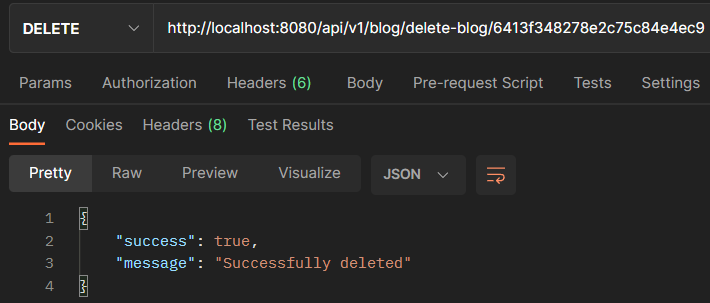
# Get single blog controller apis

* Take id through paramd and pass it into blogModel.findBuId()
* Validate if blog recive or not and send appropriate error res
* At the end send success res 
* Validate throught post man 

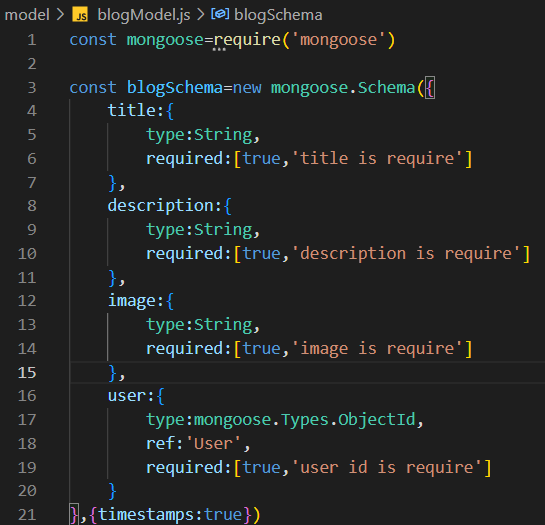
# Delete blog controller

* Follow same approach get id and pass id into blogModel.findByIdAndDelete()
* Check for validation if blog dosent found send error
* Otherwise send success res

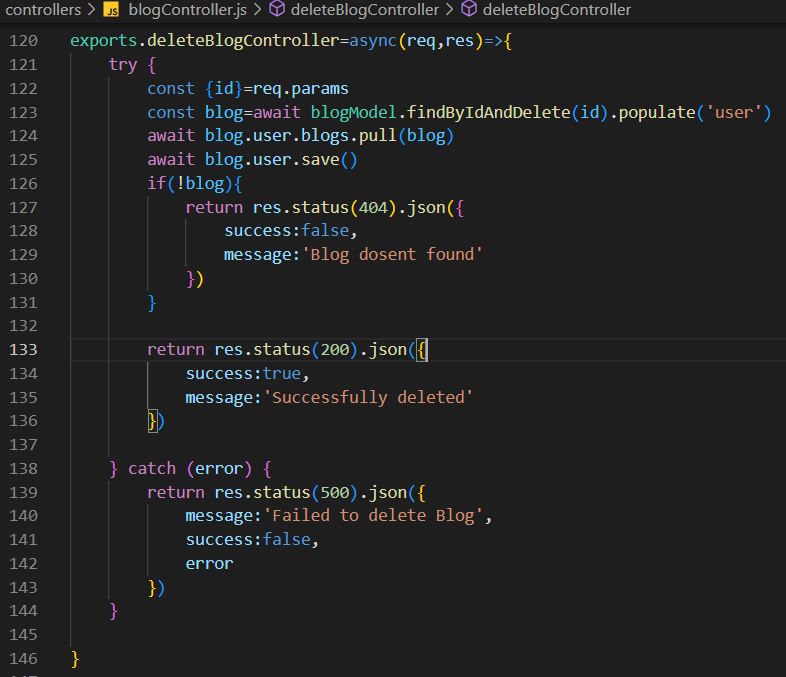


* Check into postman weather api working or not 

# Interconnect user and blog model

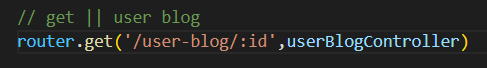
* Add blogs field inside user model and add user inside blog model  
* We are going to add user field inside blog model and while creating blog controller we will first check if user is there or not bu using userModel ,only after we will creater blog
* Now we will start session as well as start transtaction
* Now inside useModel in blogs field we are going to push our current created blog and session will be save and at the end we will commitTransaction()



* Same thing we will do while deleting blog
* If we deleting particular blog we also want we should delete it from user field from blog model also from using populate(‘user’)
* User field connected with user model which consist if blogs field array
* We will pull our current blog from blog array by using user field inside blog model
* 

# We will create user blog routes and controller

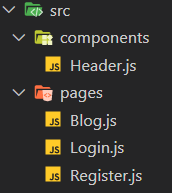
* We will follow same procedure as prev to create routes and controller



# 

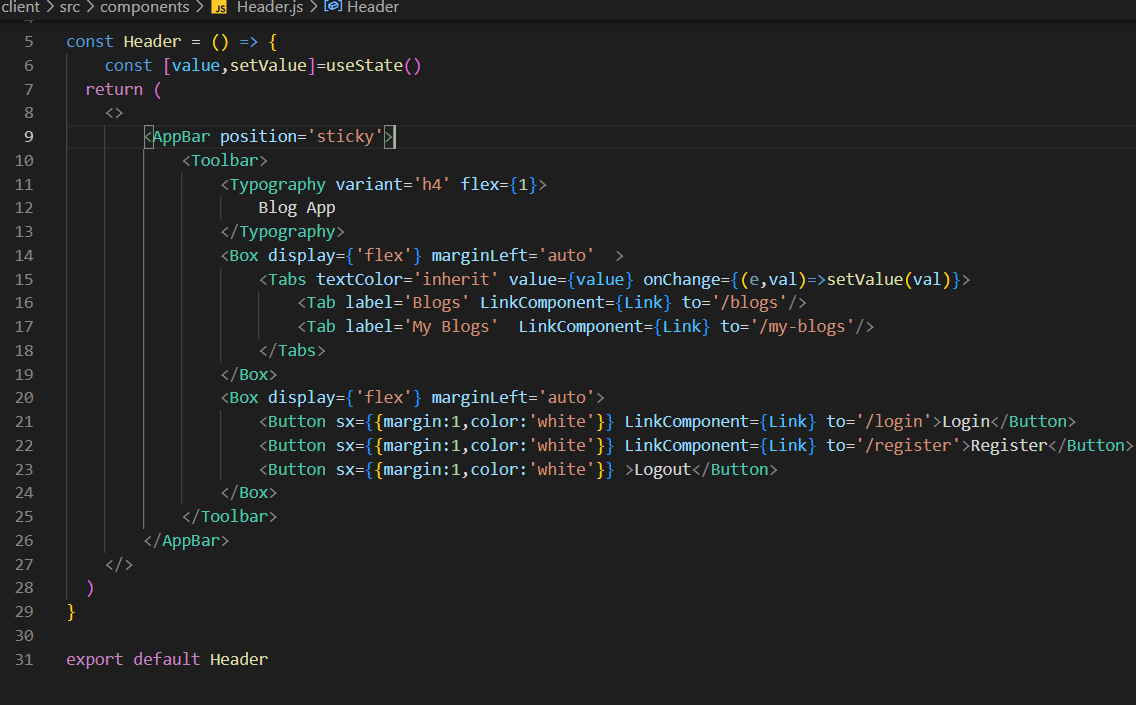
# Client side app

* Now client side app created using npx create react app
* Routing and mui dependency install 
* Now componenet and pages folder created into src
* Header file created inside component and login register blog files created inside pages

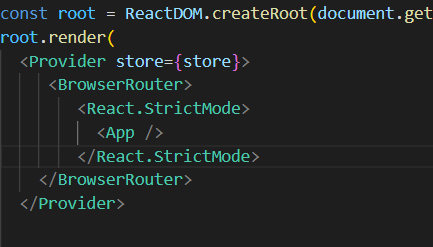


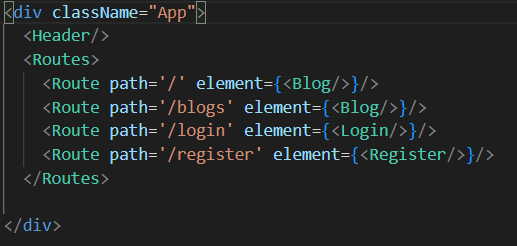
# Working on header component

* We are creating header using mui
* ImportBox,AppBar,Toolbar,Button, Typography, Tabs, Tab



# Working on routes

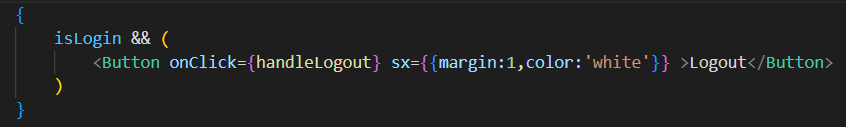
* Now create multiple pages like Login,Register,Blog
* Wrap your app component inside browserRouter
* Import routes and route in app.js file and add path and respective component inside element



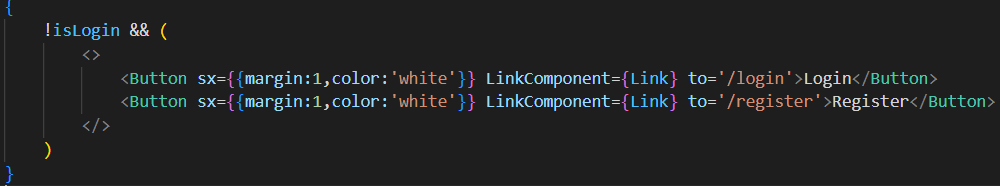
# Show diff header if user login or logout

* Add condition in header.js file if user login then show him this menu

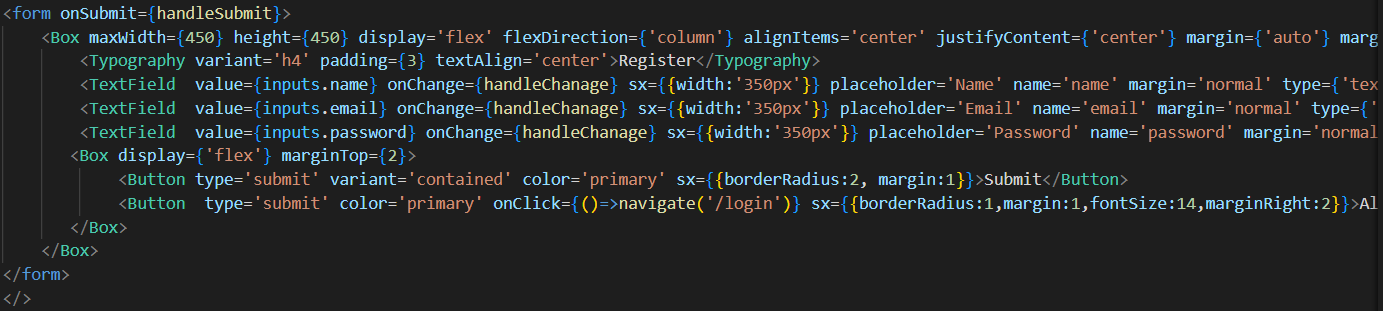
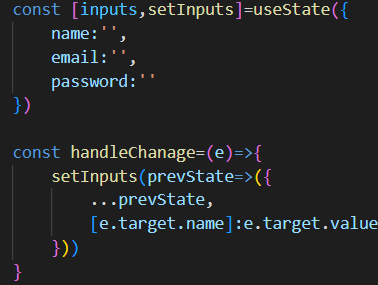


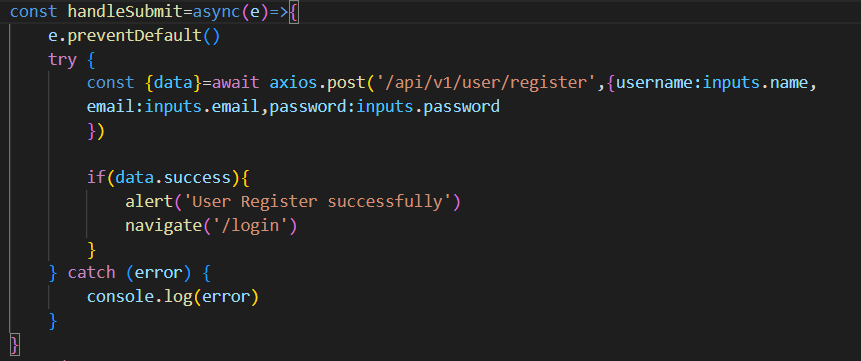


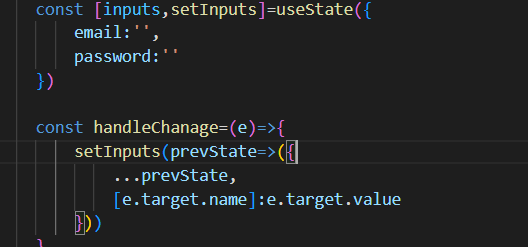
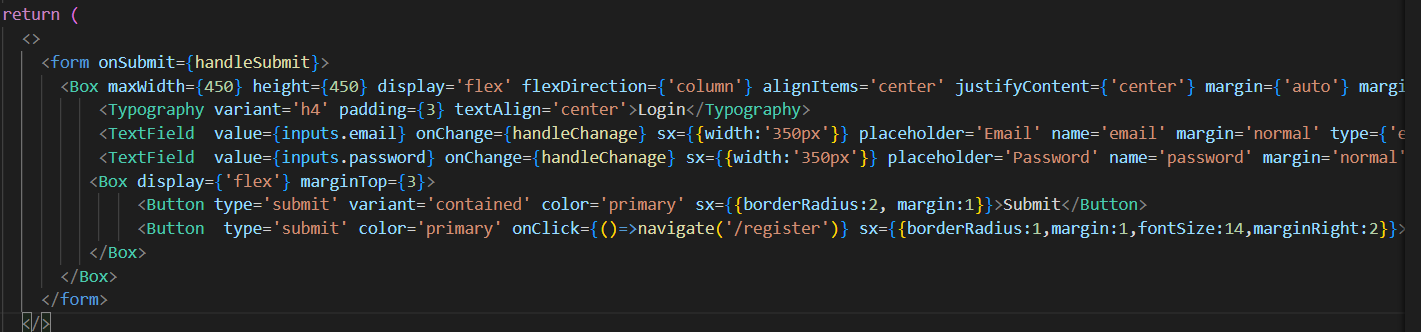
* Add condition in header.js file if user logout



# Now create login and logout component

* Import Box,typography,textInput,Button form mui
* Create state to store inputs after onchange functionality
* 
* Import axios to send api request after clicking on submit button
* Here is handle submit func where sending api req to client,then we recive response ,if res success true we dispatch action login true

And navigate ti home page using usenavigator. 

* Same we perform for login pager only diff id jus remove field of username other than that all is same
* 