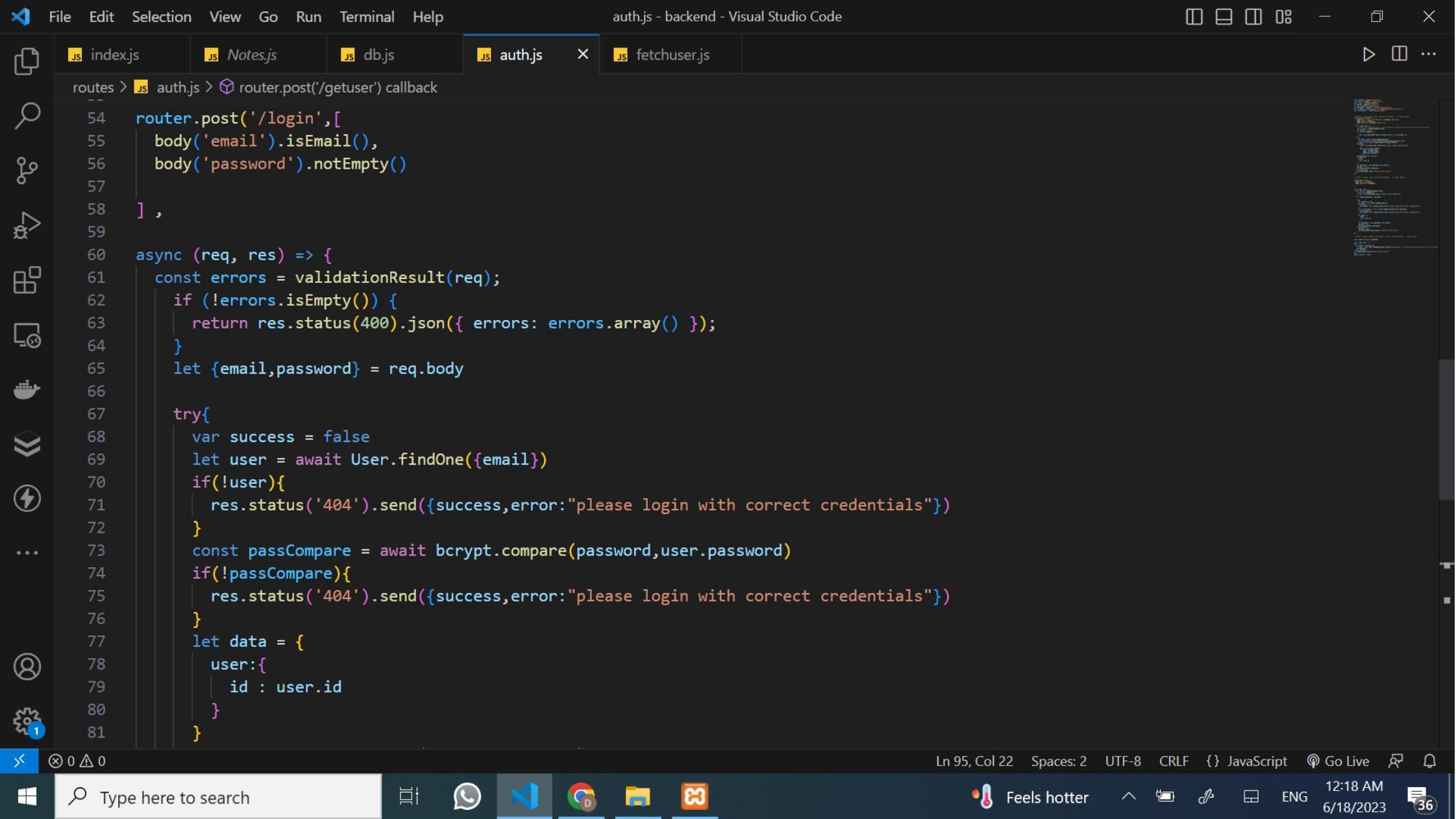
```
const express = require('express');
const User = require('../models/User')
const router = express.Router();
var bcrypt = require('bcryptjs');
var jwt = require('jsonwebtoken');
var fetchUser = require('../middleware/fetchuser');
const { body, validationResult } = require('express-validator');
const JWT_SECRET = "myNameIsDaniyalLodhi!!"



// ROUTE 1: Creating user using  /api/auth/createUser  , No login requred
router.post('/createUser',[
    body('name','name is not defined').isLength({ min: 3 }),
    body('email').isEmail(),
    body('password').isLength({ min: 5 }),
] ,
  async (req, res) => {
    // Finds the validation errors in this request and wraps them in an object with handy functions
    const errors = validationResult(req);
    var success = false;
    if (!errors.isEmpty()) {

      return res.status(400).json({ success,errors: errors.array() });
    }
    try{
      var salt = await bcrypt.genSaltSync(10);
      const hashedPass = await bcrypt.hash(req.body.password, salt )
```

```
        var salt = await bcrypt.genSaltSync(10);
        const hashedPass = await bcrypt.hash(req.body.password, salt )
      let user = await User.findOne({email:req.body.email})
      if(user){
          return res.status(400).json({error:"Email already registered"})
          }
          user = await User.create({
              name: req.body.name,
              email: req.body.email,
              password: hashedPass,
            })
      console.log("User Created")
      let data = {
        user:{
          id : user.id
        }
      }
      var authToken = jwt.sign(data, JWT_SECRET);
      success = true ;
      res.json({success,authToken})
    }  catch(error){
      res.status(500).json("internal server error")
    }}
  );

  // ROUTE 2 :(login) using  /api/auth/createUser  , No login requred

  router.post('/login',[
    body('email').isEmail(),
```


File Edit Selection View Go Run Terminal Help
auth.js - backend - Visual Studio Code
index.js
Notes.js
db.js
auth.js
fetchuser.js
routes > auth.js > router.post('/getuser') callback
router.post('/login',[
body('email').isEmail(),
body('password').notEmpty()
] ,
async (req, res) => {
const errors = validationResult(req);
if (!errors.isEmpty()) {
return res.status(400).json({ errors: errors.array() });
}
let {email,password} = req.body
try{
var success = false
let user = await User.findOne({email})
if(!user){
res.status('404').send({success,error:"please login with correct credentials"})
}
const passCompare = await bcrypt.compare(password,user.password)
if(!passCompare){
res.status('404').send({success,error:"please login with correct credentials"})
}
let data = {
user:{
id : user.id
}
}
Ln 95, Col 22 Spaces: 2 UTF-8 CRLF {} JavaScript Go Live
Type here to search
Feels hotter
ENG
12:18 AM
6/18/2023

```
77          let data = {
78            user:{
79              id : user.id
80            }
81          }
82          var authToken = jwt.sign(data, JWT_SECRET);
83          success = true
84          res.json({success,authToken})
85        }catch(error){
86          success = false
87          res.status(500).json(success,"internal server error")
88        }
89    })
90
91    // ROUTE 3 :geting loggedin user details  using  /api/auth/getuser  , login requred
92
93    router.post('/getuser',fetchUser ,
94
95    async (req, res) => {
96    try {
97      let userId = req.user.id
98      let user = await User.findById(userId).select("-password") /* "-password"so password should not be provi
99      res.send(user)
100   } catch (error) {
101     es.status(500).json("internal server error")
102   }})
103   module.exports = router
104
```

index.js | db.js | auth.js | notes.js | fetchuser.js

routes > notes.js > ...

```
const express = require('express')
const router = express.Router();
const Notes = require('../models/Notes')
const fetchUser = require('../middleware/fetchuser')
const { body, validationResult } = require('express-validator');

// ROUTE 1 :get all the notes using  /api/auth/fetchallnotes  , login requred
router.get('/fetchallnotes', fetchUser, async (req, res) => {

    const notes = await Notes.find({ user: req.user.id });
     res.json(notes);
})


// ROUTE 2 : create Note  /api/auth/createnote  , login requred
router.post('/createnote',fetchUser,[
    body('title','Enter a valid title').isLength({ min: 3 }),
    body('description').isLength({ min: 5 }),
] , async (req,res)=>{
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.status(400).json({ errors: errors.array() });
    }
    try {
        const {title, description ,tag} = req.body
        const note = new Notes({title, description ,tag,user:req.user.id})
        const savedNote = await note.save()
        res.json(req.user.id)
    } catch (error) {
        res.send(error)
    }
})
```

index.js | db.js | auth.js | notes.js | fetchuser.js

routes > notes.js > ...

```
33
34  // ROUTE 3 : update Note  /api/auth/updatenote/:id  , login requred
35
36  router.put('/updatenote/:id',fetchUser, async (req,res)=>{
37      // let id = req.header('id')
38      let id = req.params.id
39      let note = await Notes.findById(id)
40      const {title, description ,tag} = req.body
41      let newNote = {}
42      if(title)newNote.title = title
43      if(description)newNote.description = description
44      if(tag)newNote.tag = tag
45
46      try {
47          if (note.user.toString() !== req.user.id){
48              return res.status(401).send("not allowed to make changes")
49          }
50          else{
51          note = await Notes.findByIdAndUpdate(id,{$set:newNote},{new:true})
52          // res.send("author matched , Changes have been made")
53          res.send(note)
54      }
55      } catch (error) {
56          res.send(error)
57      }
58  })
59  // ROUTE 4 : delete Note  /api/auth/deletenote/:id  , login requred
60
61  router.delete('/deletenote/:id',fetchUser, async (req,res)=>{
62      // let id = req.header('id')
63      let id = req.params.id
64      let note = await Notes.findById(id)
```

```
63      let id = req.params.id
64      let note = await Notes.findById(id)
65      if(!note){return res.status(404).send("note note found")}
66
67      try {
68          if (note.user.toString() !== req.user.id){
69              return res.status(401).send("not allowed to make changes")
70          }
71          else{
72          note.deleteOne()
73          res.send("note deleted")
74      }
75      } catch (error) {
76          res.send(error)
77      }
78  })
79  module.exports = router ;
```