Lesson 1 :-EXPRESS API

**Index**

* INTRODUCTION OF EXPRESS API
* PACKAGES WITH INSTALL COMMAND
* CREATE JS FILES AND MAKE CONNECTION
* USE POSTMAN FOR SEND REQUEST
* DEBUGGING

*1.INTRODUCTION OF EXPRESS API*

***Express API*** *is a web API that uses the* ***RESTful approach.******REST API*** *is an API that is use* ***resource-based interfaces****. In EXPRESS API that use web resources that’s is formatted in JSON or XML file that accessed via HTTP represented by URL (Path).*

* ***Express: -*** *Express is a framework in express.js .It provides various features on web application. For Installing express use*

**npm i express.**

* ***Mongoose –*** *Mongoose provided database access for the local web site. It provides a schema that’s connecting to mongo DB database. For installing mongoose use*

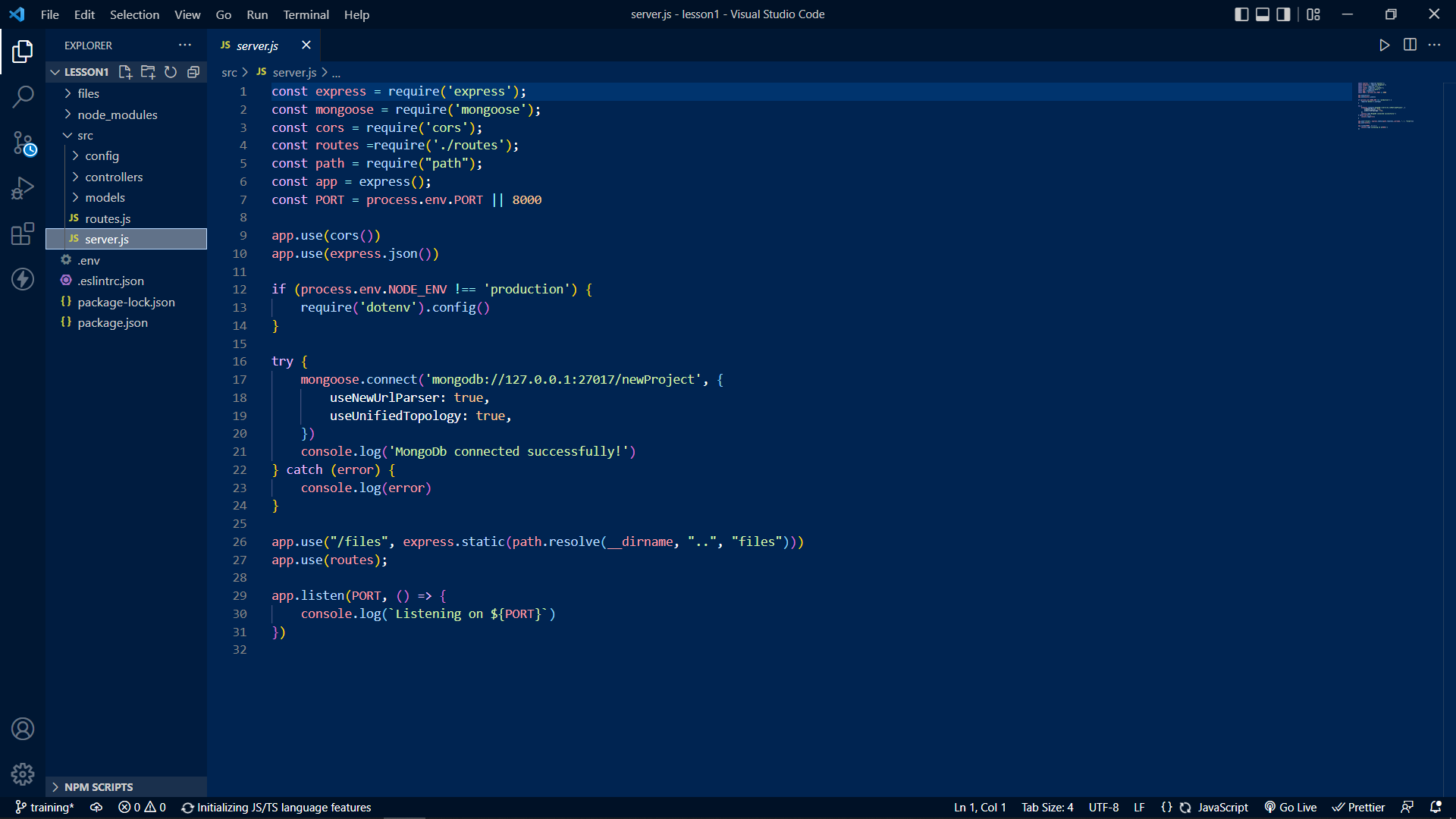
**npm i mongoose**

* ***Multer-*** *It is use in node JS middleware multipart or form data which is primary used. Its function that received the request and send response object when user create any type of request. For installing multer package we use the command*

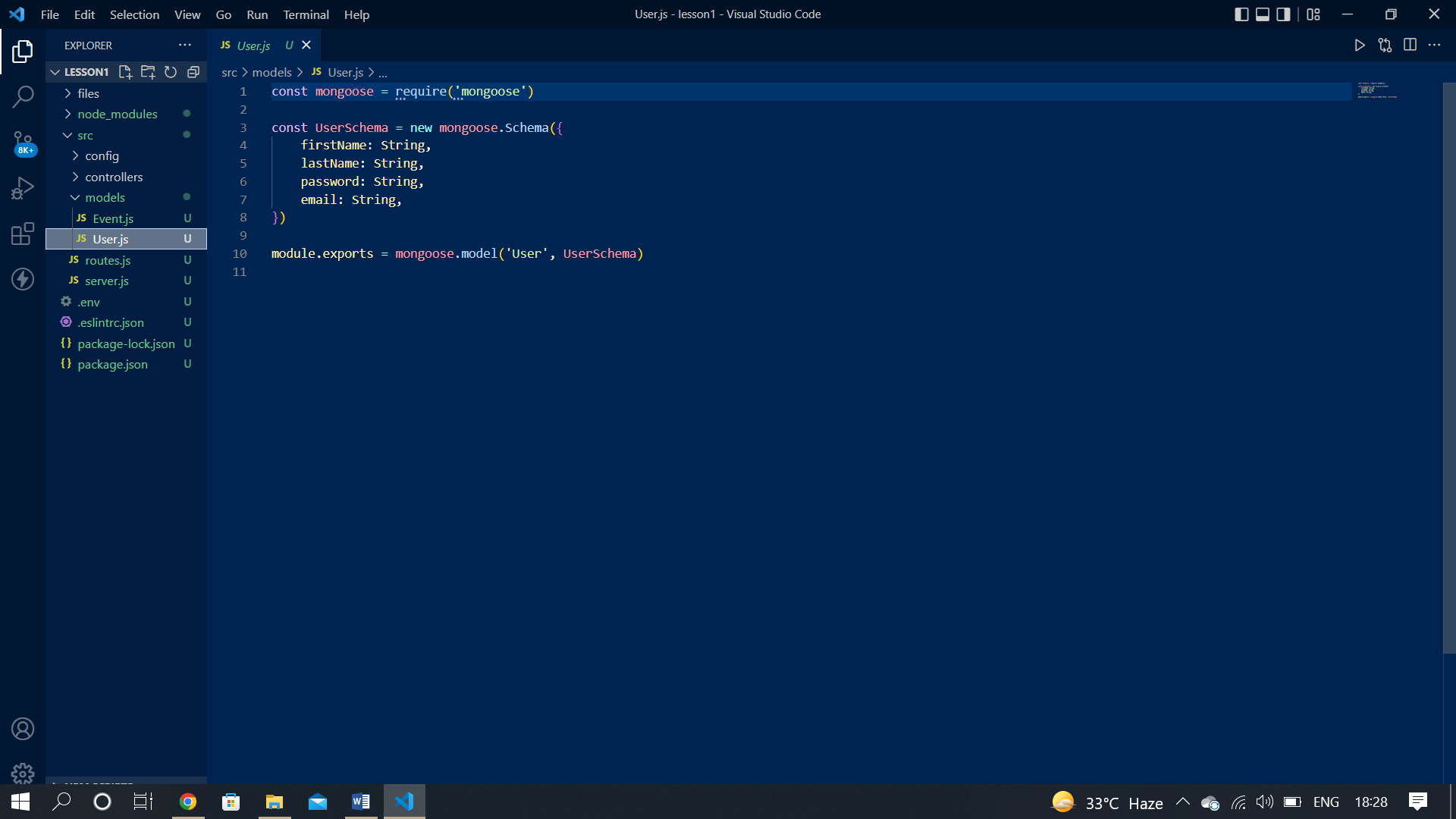
**npm i multer.**

**3. CREATE JS FILES AND MAKE CONNECTION**

*Creating Server.js file*

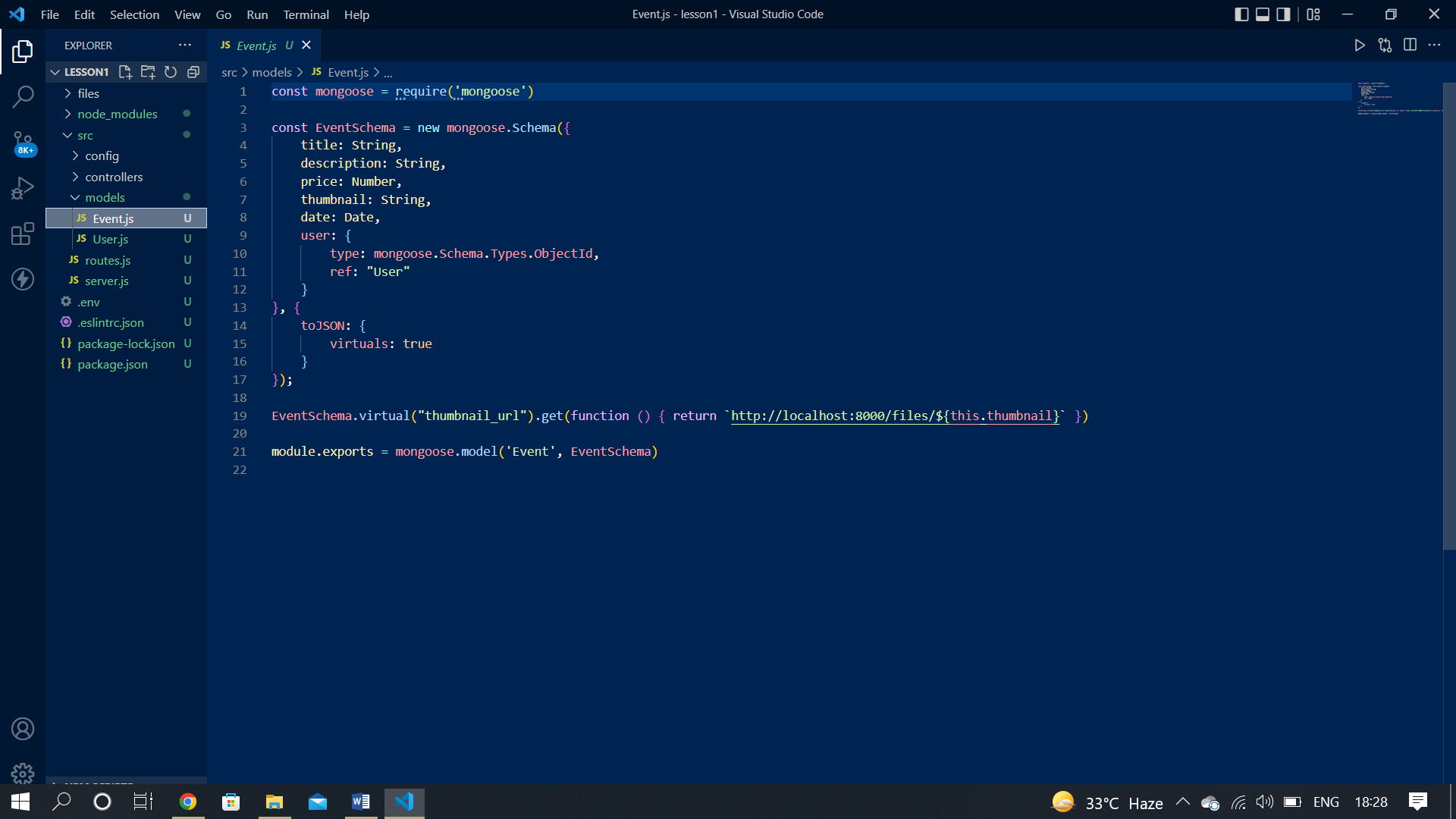


*Creating user.js File*

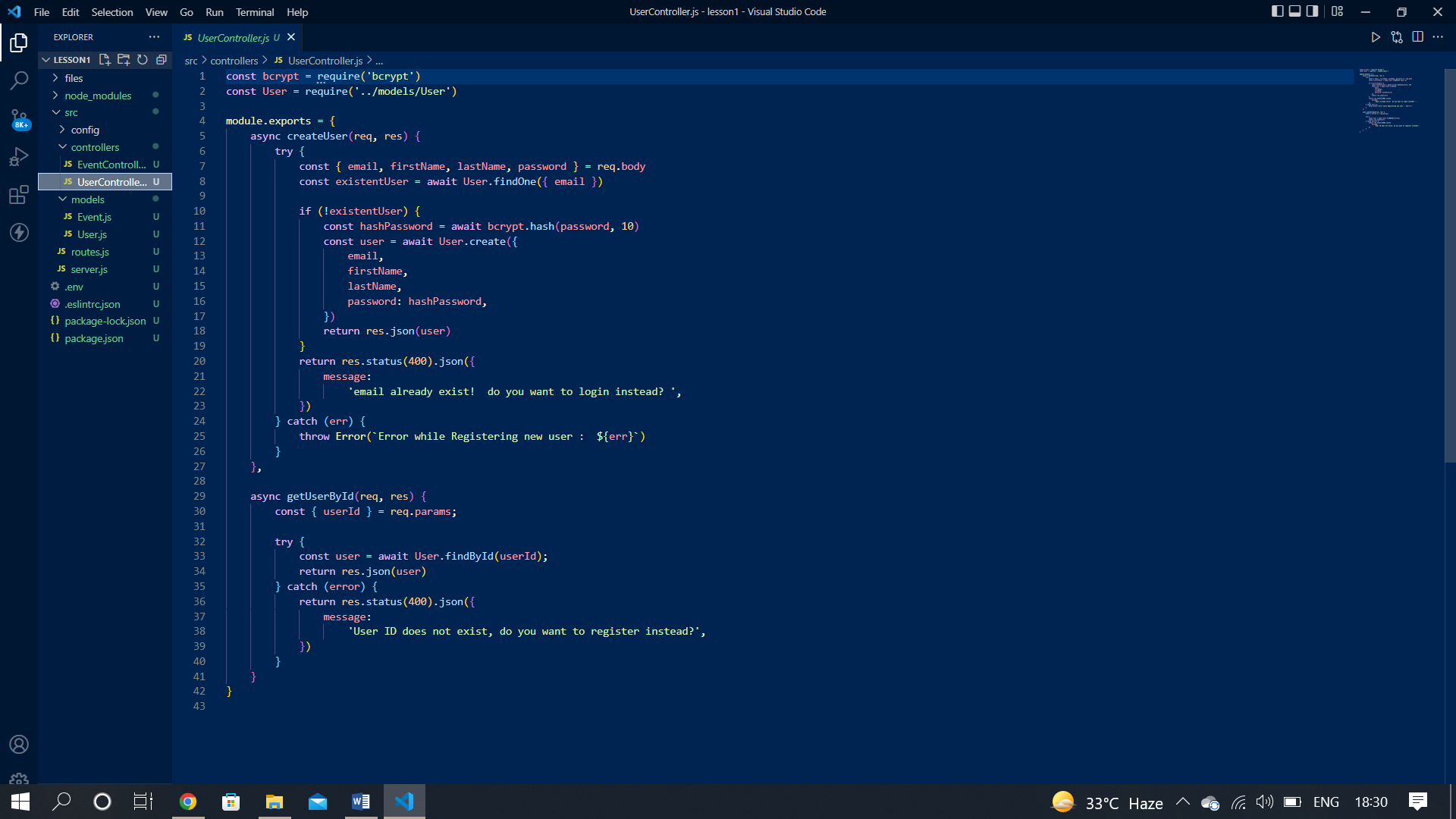


Create another file in **“models/event.js”.**

*Create event.js*



*UserController.js*

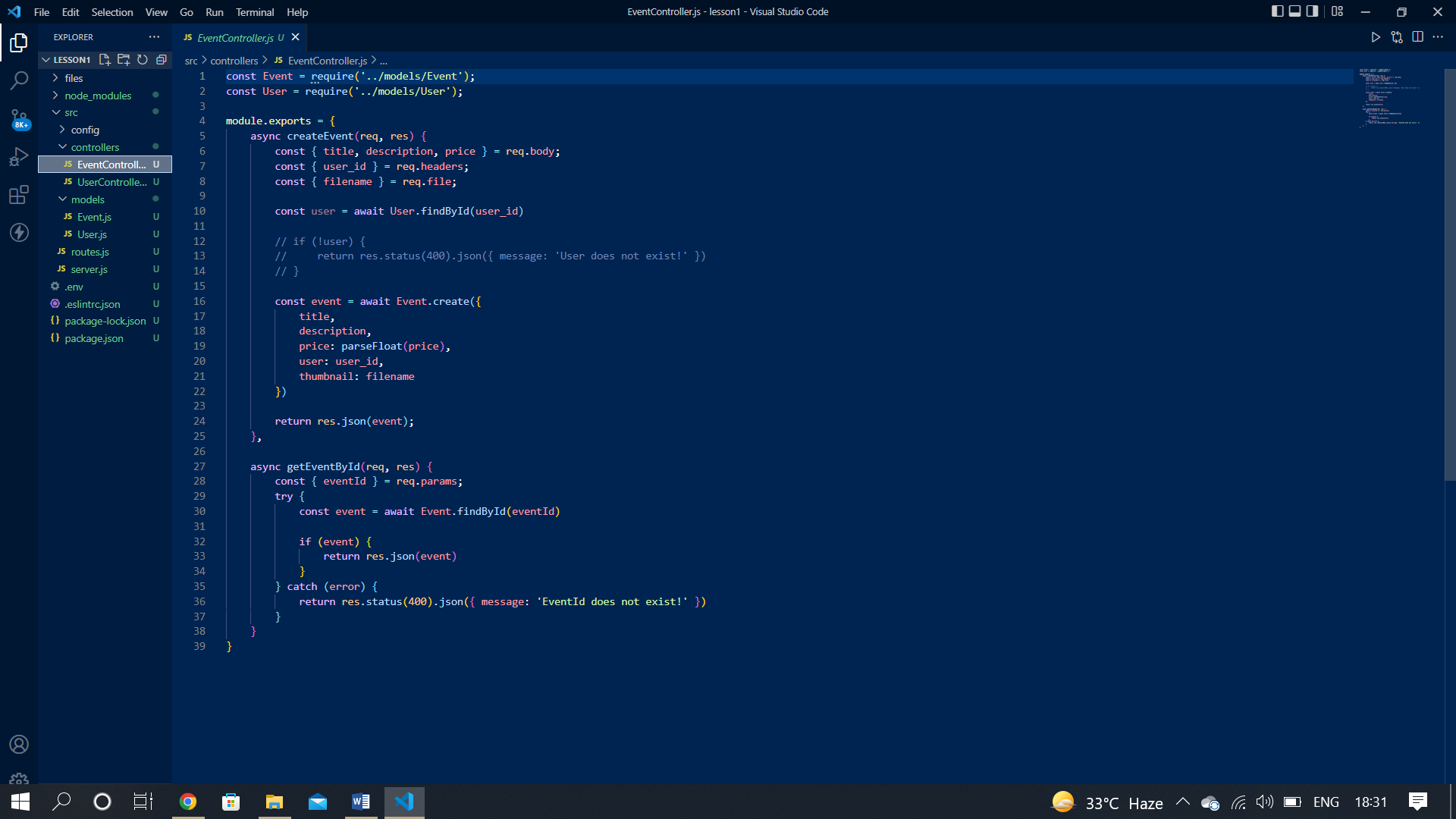


*In the UserController.js file we are control or call the user sub file. First we export the module and fetch that user is exist on the database or not if exist then show a massage user is exist and if not exist then create a new user using user schema.*

*After that we use* ***getUserById () method*** *and use the User\_id and fetch the user is exist on database or not if exist it show the result and not then 400 error massage show in console log.*

In controller folder, add another file **“EventController”.**

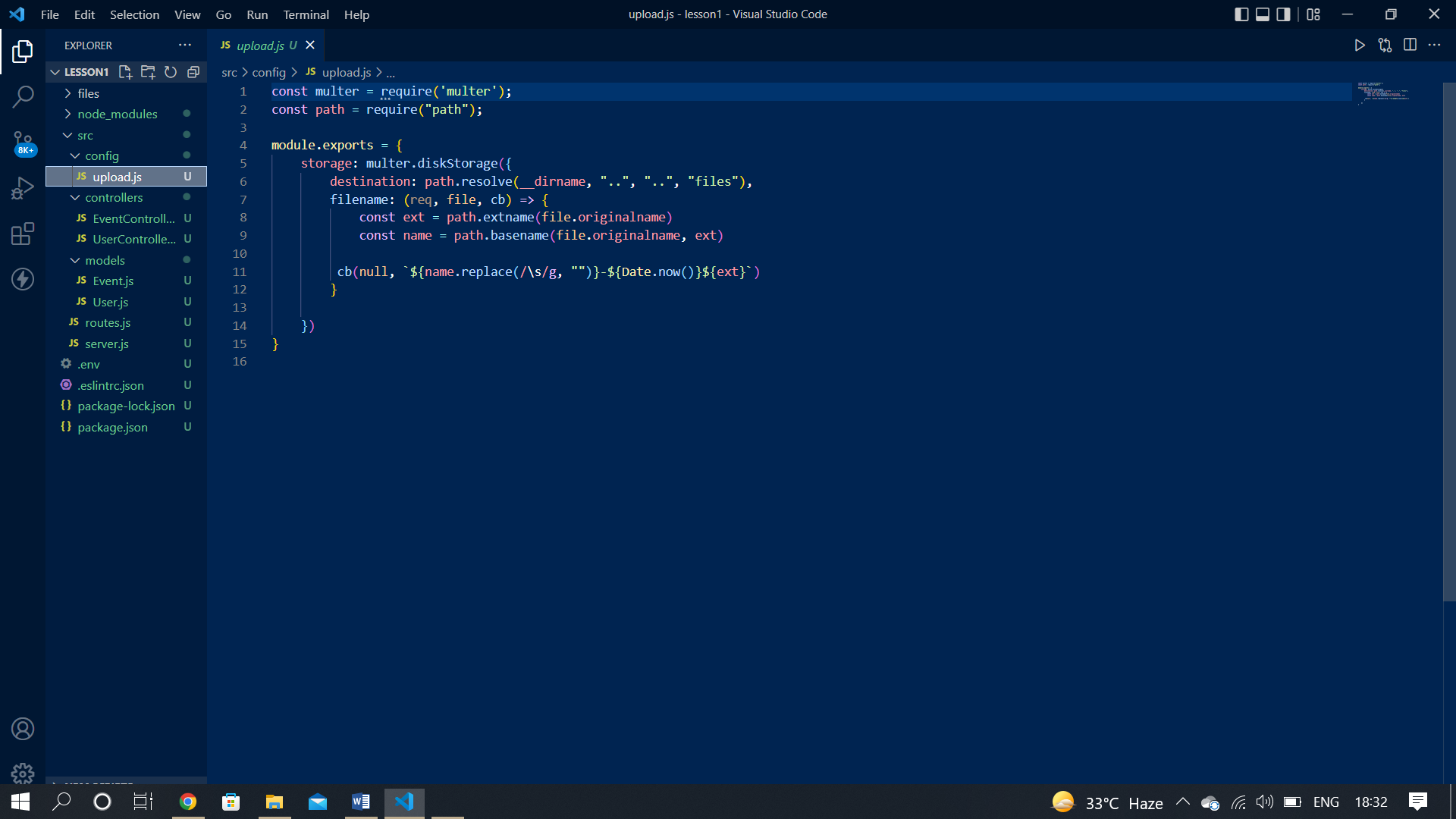
**EventController.js**



*In the Eventcontroller.js file we are control or call event sub file. First we create event to receive all the information from the request. Then we apply* ***findById() method*** *to create user and check it using User\_id.If the user is exist then it create an event or if not exist then it show a massage user is not exist.*

*Create folder in src name it as* ***“config”.*** *In this folder create one new file* ***“upload.js”.***

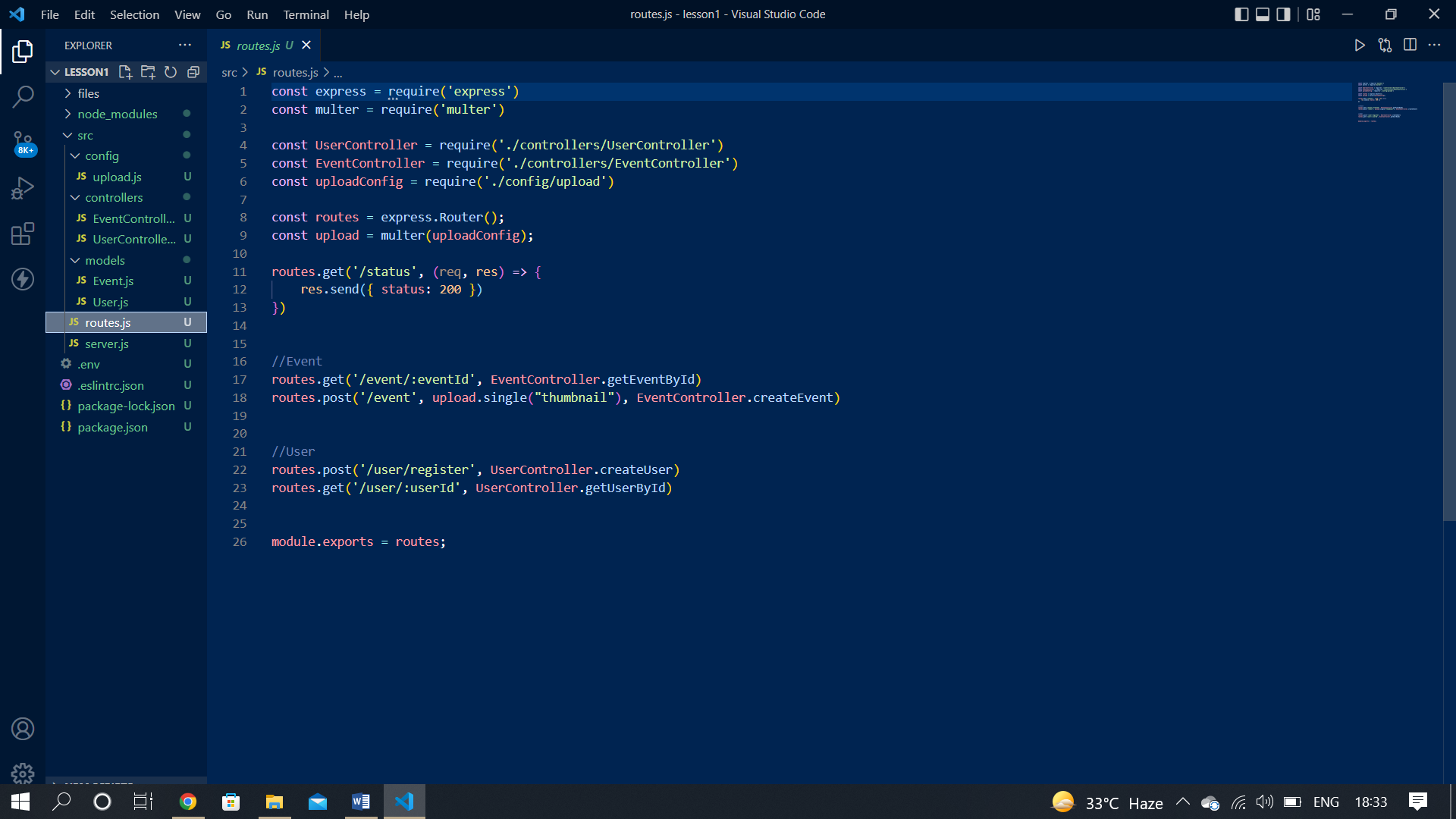
**upload.js**



*In the upload.js file we construct the file path. First we construct the path and then storage in local and make and URL. After that it’s back to controller with file extension. If that file had any rejection then its show error massage using controller file.*

*Now make one new file in src and named it as****"routes.js"****. We use it as middleware to allow import the path of different files.*

**routes.js**



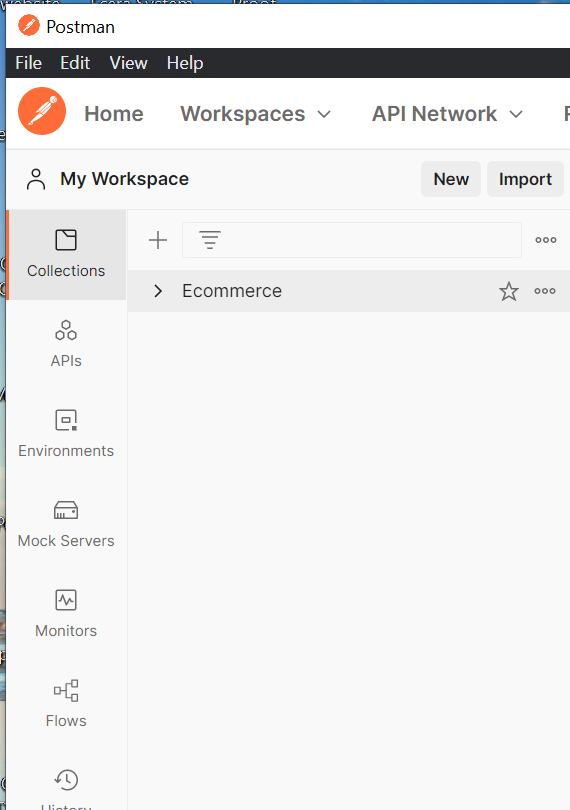
*In the routes.js file we are connected all the middleware that are routes of the main file . And also create the user and event routes that we call through an API.*

*POSTMAN*

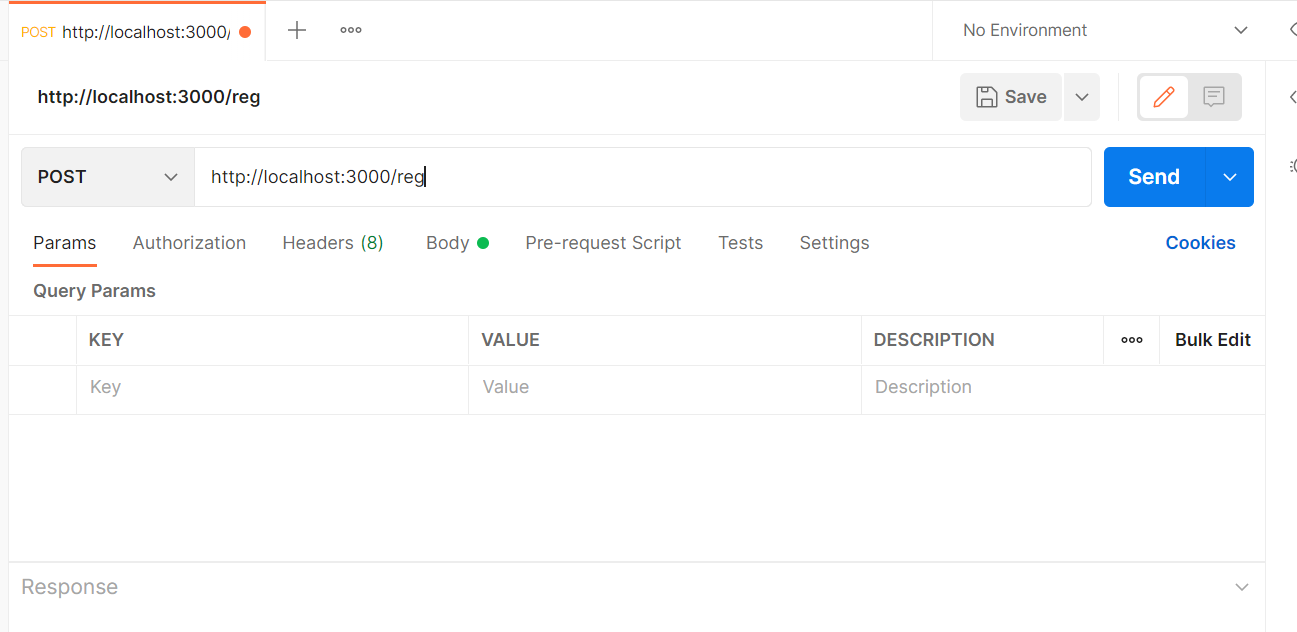
*It is a software testing tool to test APIs. With this we can create, test, share and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses. It is used for backend testing where we enter the end-point URL, it sends the request for the server and receives the response back from server.*

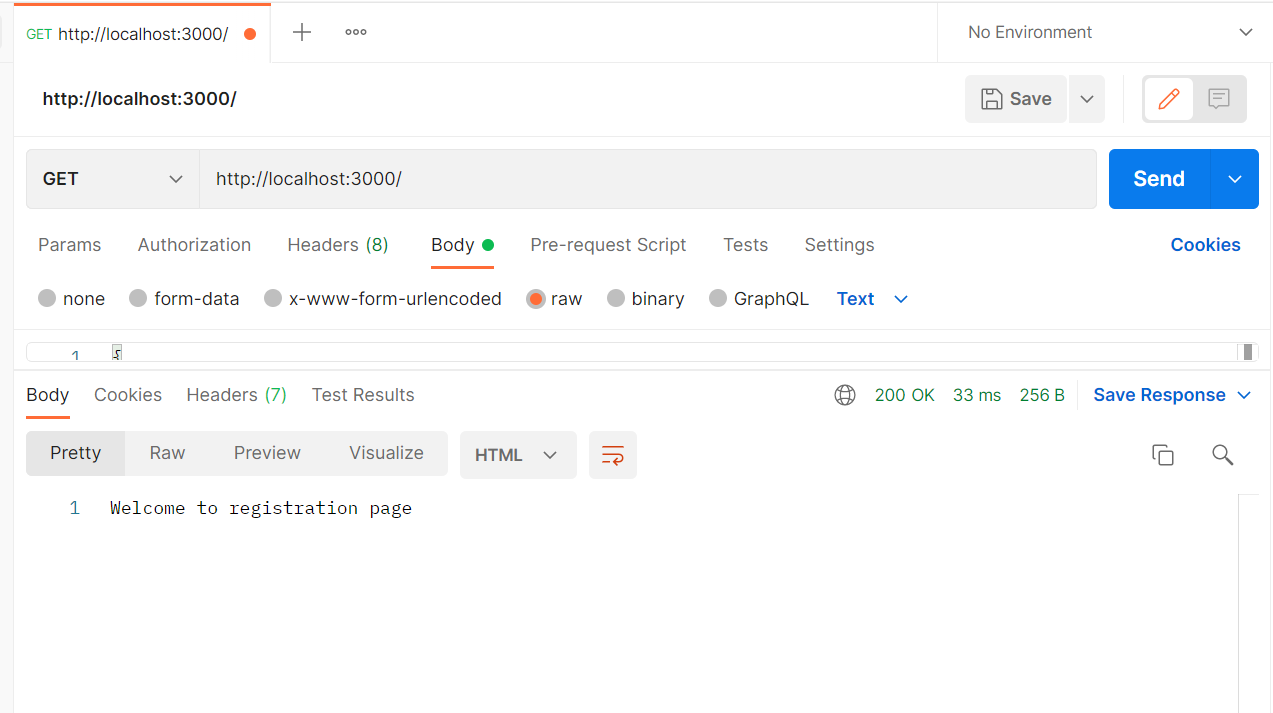
*How we use POSTMAN*

*Step 1:- Go to the collection tab click on the ‘+’ button create a new collection it will appear name or other setting.*

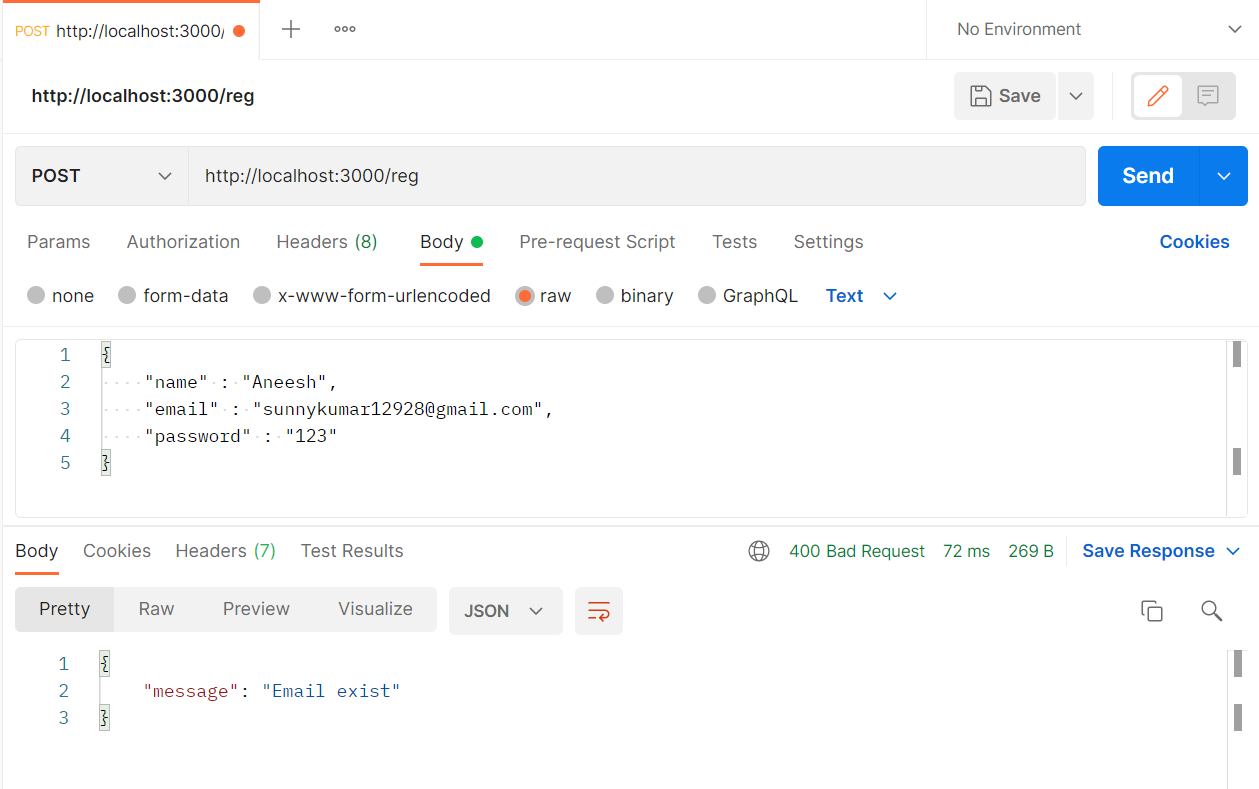


*Step 2:- Then select create a new request and it will open a window on post man application like this.*

*Step 3 :- Set the address of your local host and add the request type get and check the result. If it shows status code 200 in green then it is ok.*



Step 4 - Set post request and put the value and check the status if it show 200 then it’s ok or if show 404 then error is process.



*INSOMNIA*

*It is a free cross-platform desktop application that takes the pain out of interacting with and designing HTTP-based APIs. It combines an easy-to-use interface with advanced functionality like authentication helpers, code generation, and environment variables.*

*6. DEBUGGING*

*The Visual Studio Code editor has all the tools to debug Node.js applications effectively. Its built-in debugger can debug any application that targets the Node.js runtime, even if the source code for the application is a language that transpires to JavaScript, such as TypeScript.*

*The easiest way to start a debugging session in Visual Studio Code is to open a file in the editor, click the Run View icon in the Activity Bar (or press Ctrl+Shift+D on your keyboard), followed by the Run and Debug button at the top left corner of the application.*

*The Visual Studio Code debugger will try to auto-detect the debug environment for your project, but if this fails, you will be prompted to select the appropriate environment; in this case, select Node.js.*

*After selecting an environment, the project launches and the debugger attaches to the process. You can see the output of your project in the DEBUG CONSOLE, and the debug toolbar appears at the top of the screen to step through the code, pause the script, or end the session. On the left-hand side of the editor, there are five panes titled VARIABLES, WATCH, CALL STACK, LOADED SCRIPTS, and BREAKPOINTS.*