Experiment 09

# **Aim:** Create the back-end application using Node.js or Express (Callbacks, Event loops).

# **Requirements:** Chrome, VsCode

## **Theory:**

## **ExpressJS:** .

# Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

# Allows to set up middlewares to respond to HTTP Requests.

# Defines a routing table which is used to perform different actions based on HTTP Method and URL.

# Allows to dynamically render HTML Pages based on passing arguments to templates.

# Installing Express

# Firstly, install the Express framework globally using NPM so that it can be used to create a web application using node terminal.

# $ npm install express –save

# The above command saves the installation locally in the node\_modules directory and creates a directory express inside node\_modules. You should install the following important modules along with express −

# body-parser − This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.

# cookie-parser − Parse Cookie header and populate req.cookies with an object keyed by the cookie names.

# multer − This is a node.js middleware for handling multipart/form-data.

# $ npm install body-parser --save

# $ npm install cookie-parser --save

# $ npm install multer --save

# **Callback function:**

# Callback is an asynchronous equivalent for a function. A callback function is called at the completion of a given task. Node makes heavy use of callbacks. All the APIs of Node are written in such a way that they support callbacks.

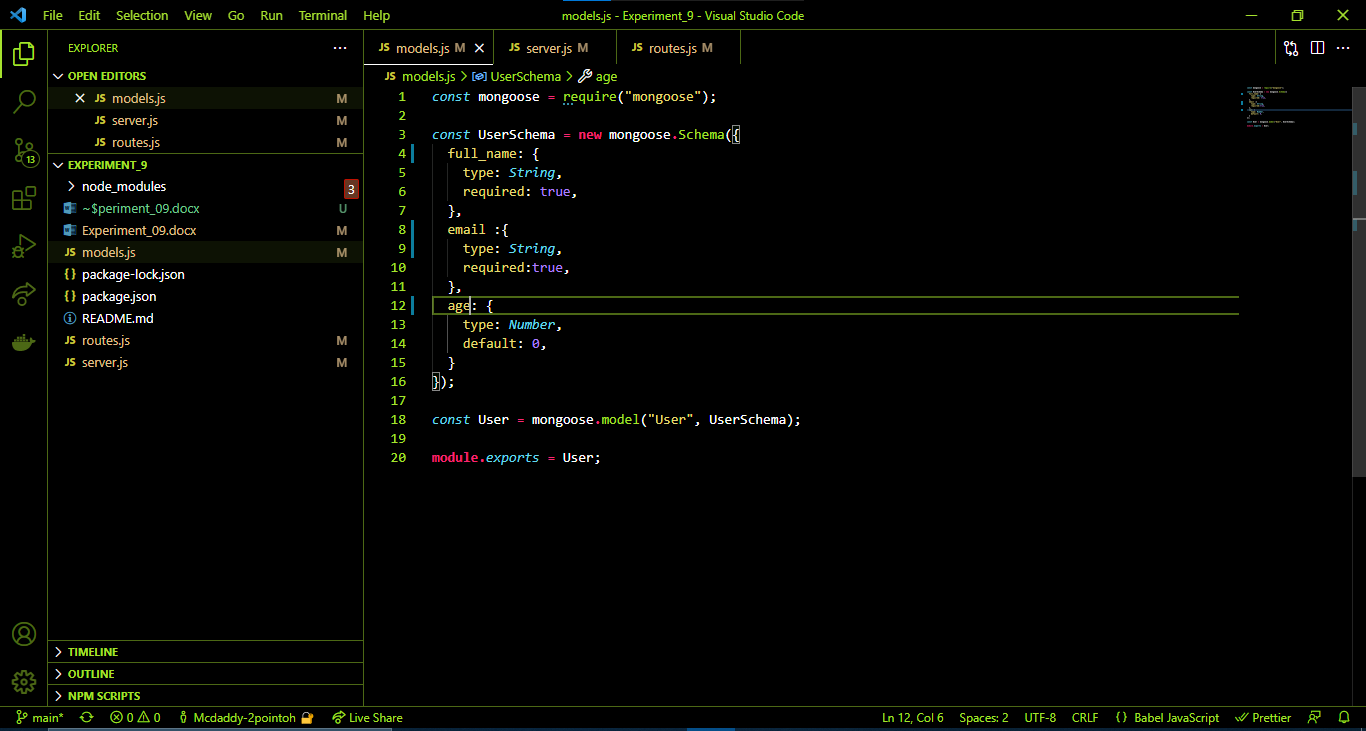
# For example, a function to read a file may start reading file and return the control to the execution environment immediately so that the next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as a parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process a high number of requests without waiting for any function to return results.

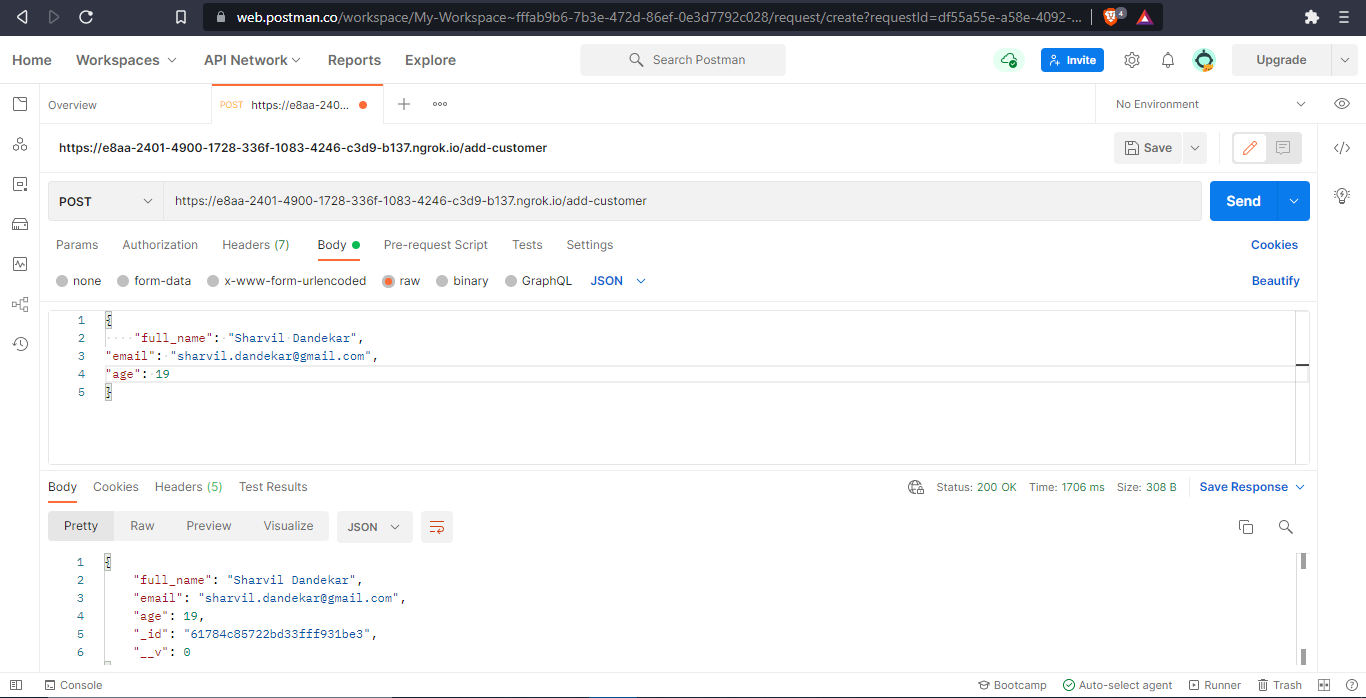
# **Event Loops:**

The event loop is what allows Node.js to perform non-blocking I/O operations — despite the fact that JavaScript is single-threaded — by offloading operations to the system kernel whenever possible.

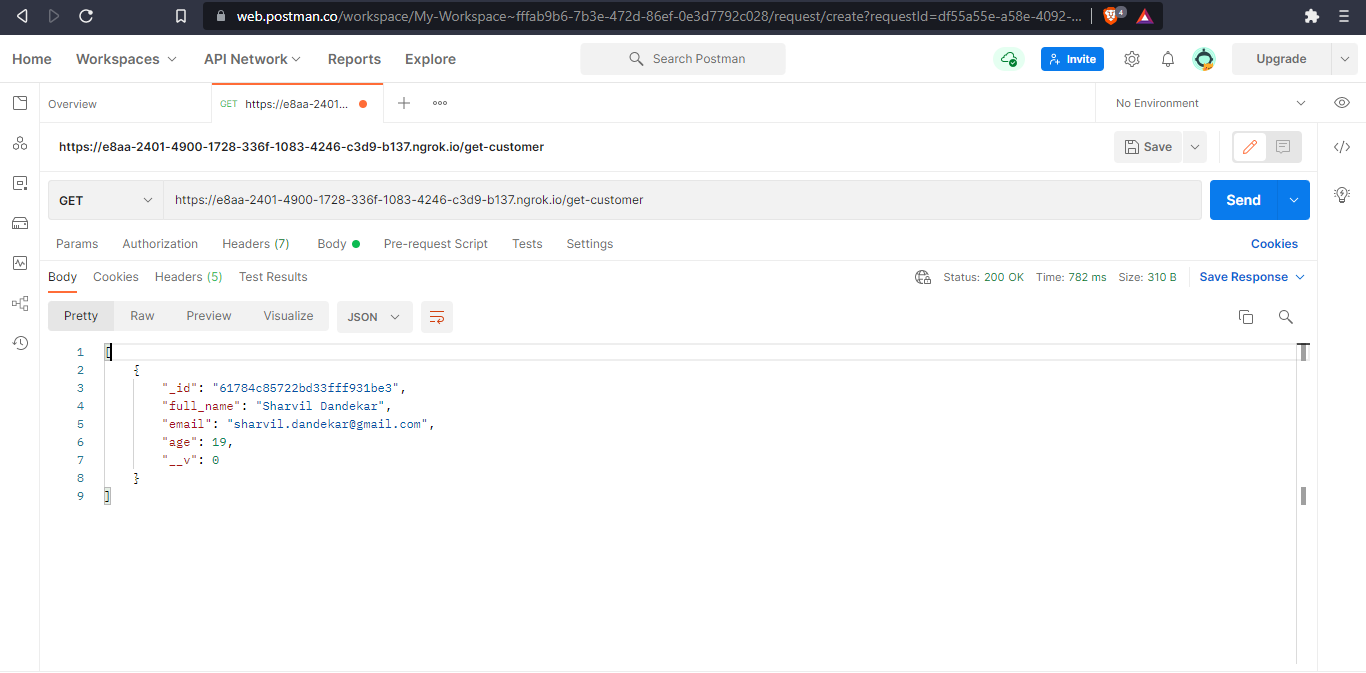
Since most modern kernels are multi-threaded, they can handle multiple operations executing in the background. When one of these operations completes, the kernel tells Node.js so that the appropriate callback may be added to the **poll** queue to eventually be executed. We'll explain this in further detail later in this topic.

# **Output:**





[post req]



[get req]

# **Conclusion:** By using express js we can make APIs. In express js we can add call back functions like middleware functions for authentication which returns after user is authenticated and passes the job to next function after middleware function. It is also easy to setup.

**References**: <https://www.w3schools.com>