**Week – 5 & 6**

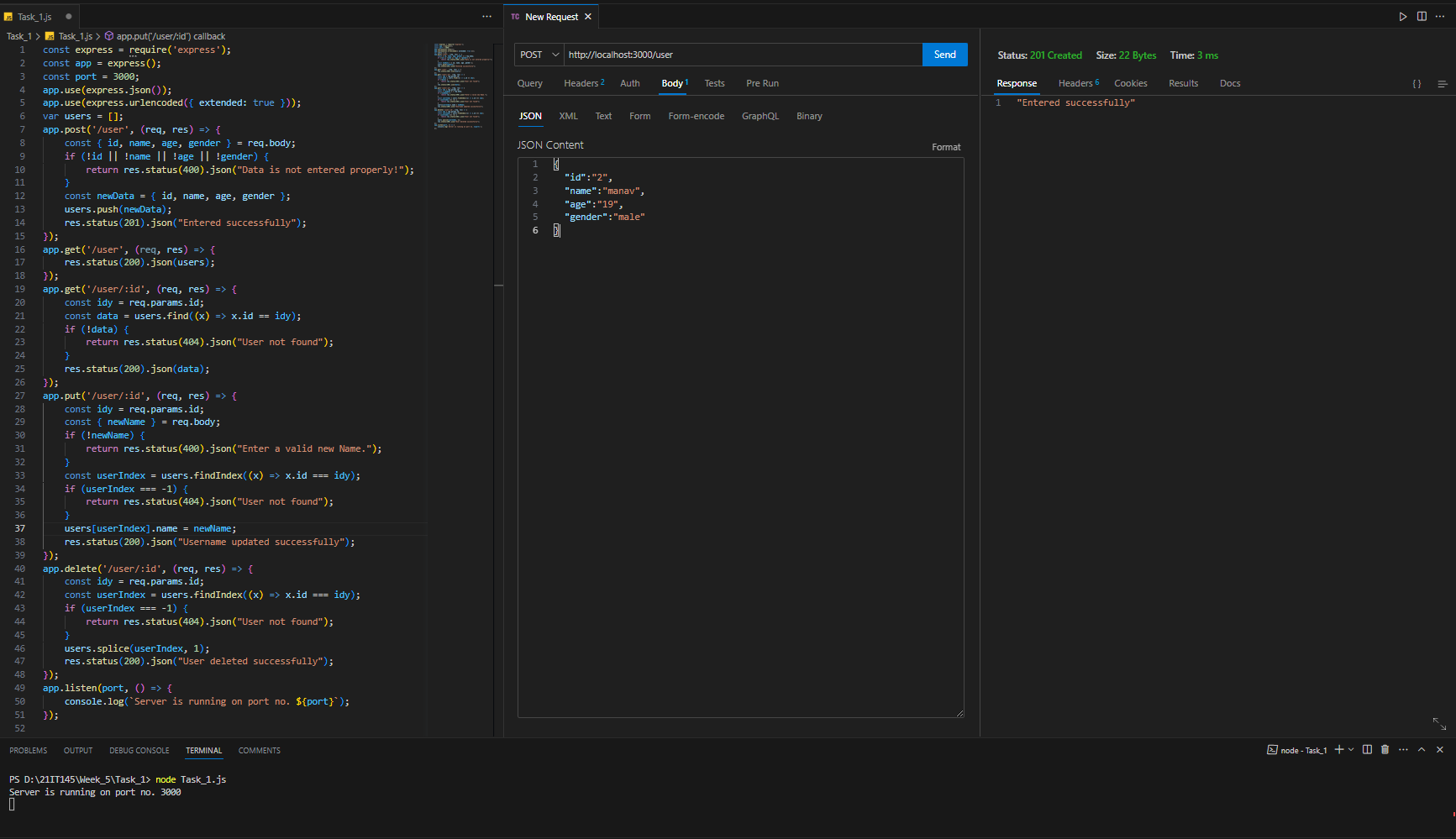
**Task-1**

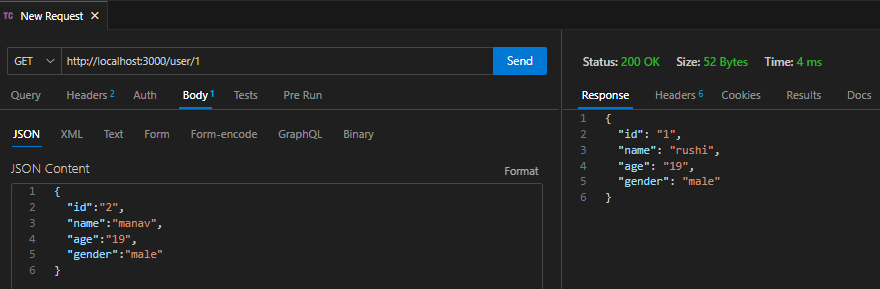
**Aim: Develop an ExpressJS API to perform CRUD operations using In Memory Database & Test it.**

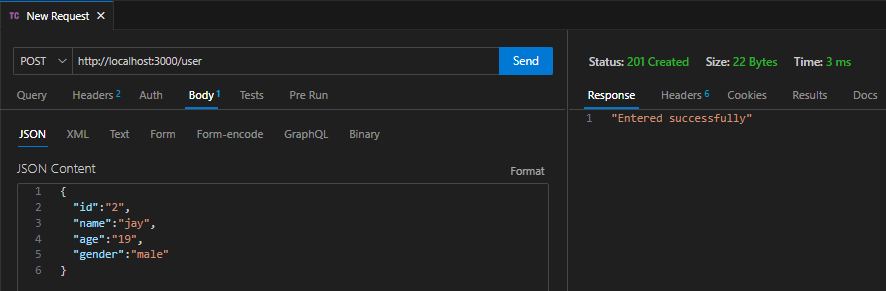
**Description:**

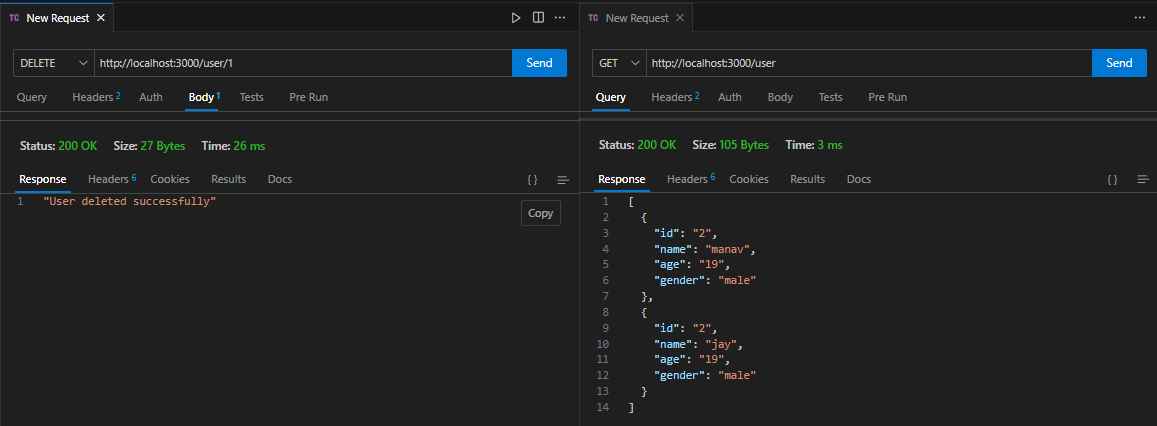
* In Express JS, CRUD refers to the four basic operations that can be performed on a database or resource - Create, Read, Update, and Delete.
* ExpressJS API to perform CRUD (Create, Read, Update, Delete) operations using an in-memory database.
* Use a tool like Postman or a web browser to test the API endpoints. Test each CRUD operation to ensure they work correctly.

**Source Code & Output:**

****

****

****

****

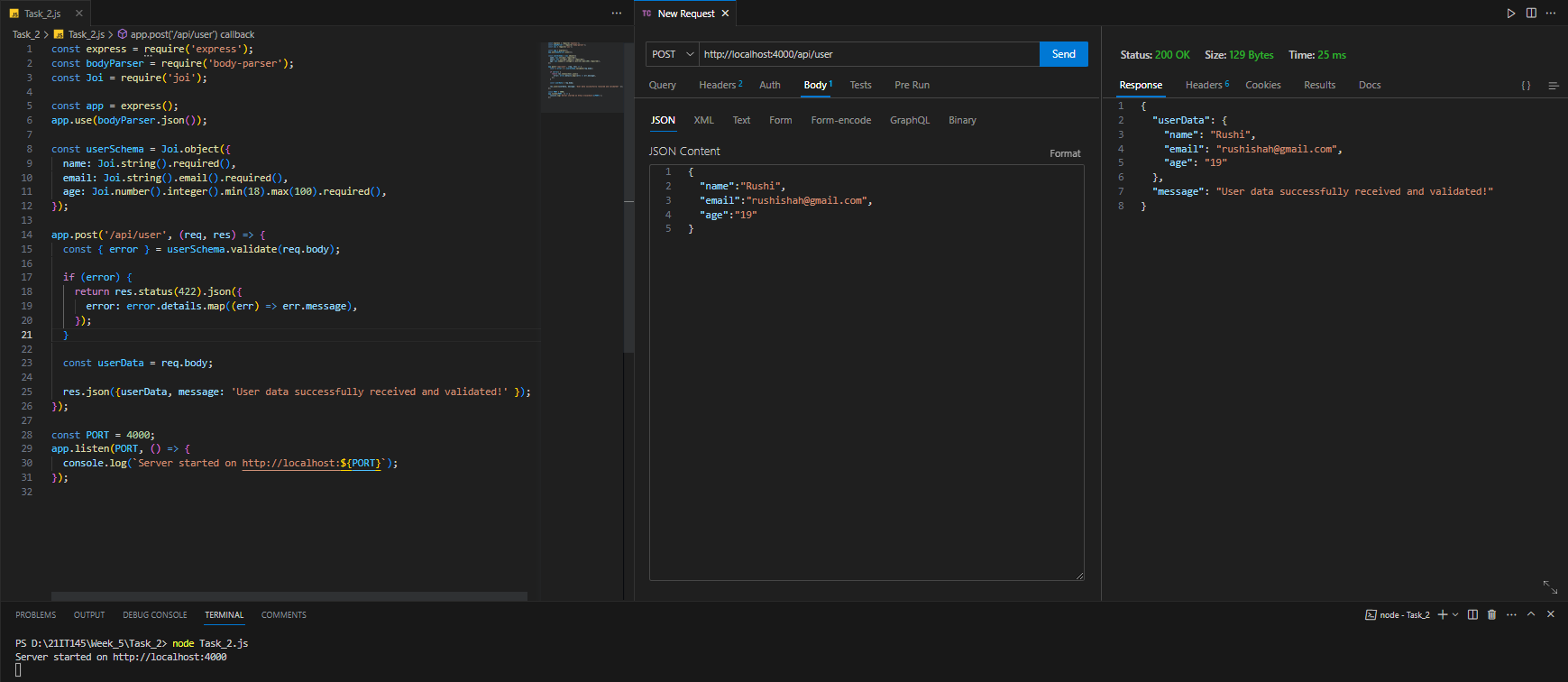
**Task-2**

**Aim: Validate API-Form/Data at server-side using validator/joi module**

**Description:**

* Joi module is a popular module for data validation. This module validates the data based on schemas. There are various functions like optional(), required(), min(), max(), etc which make it easy to use and a user-friendly module for validating the data.
* It allows you to define a schema that represents the expected structure and constraints of your data.
* Schema: A schema is an object that represents the rules and constraints applied to the data. It defines the allowed keys, data types, lengths, formats, and any other validations needed for each field.
* Validation: The process of checking the input data against the defined schema. If the data meets the schema's requirements, it is considered valid otherwise validation errors are generated.
* Sanitization: Joi can also handle data sanitization, which involves modifying the input data to ensure it adheres to the desired format, such as trimming extra spaces or converting strings to lowercase.

**Source Code & Output:**

****

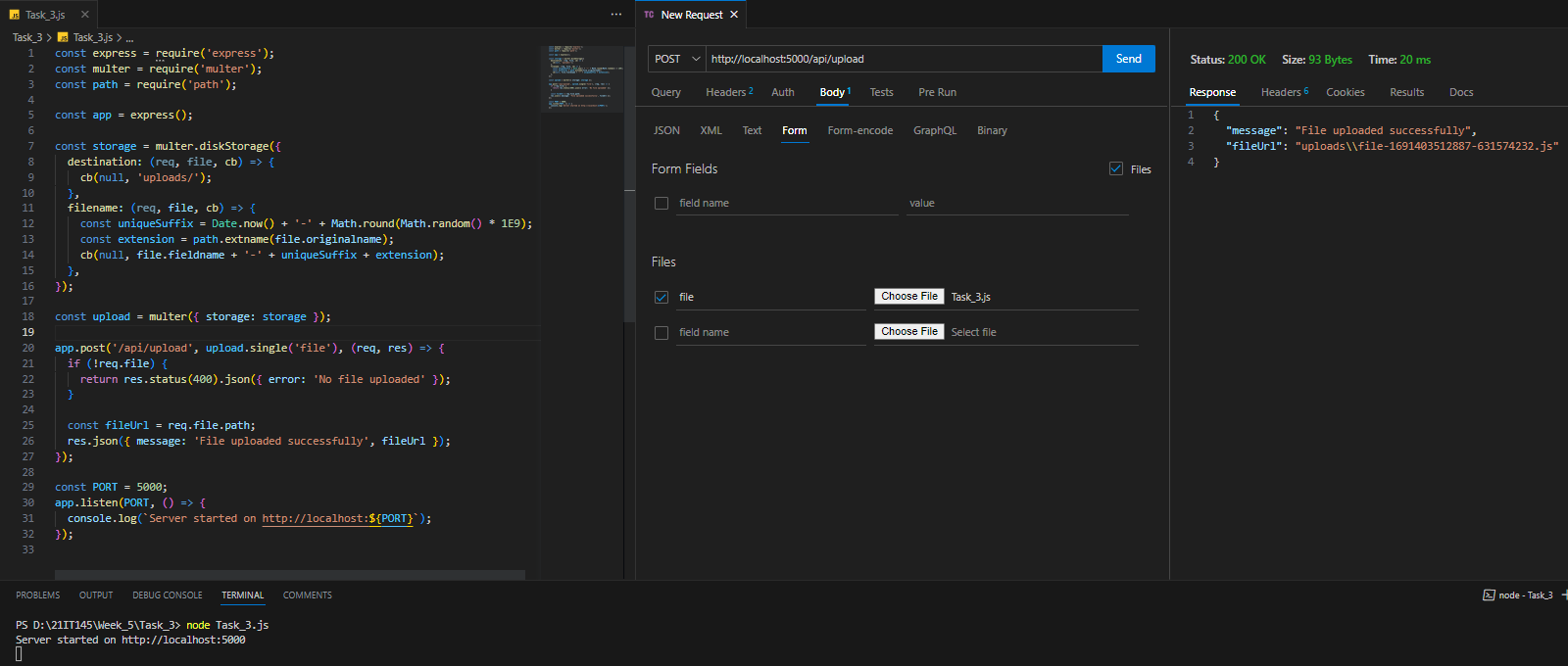
**Task-3**

**Aim: Create API to upload file & Test it.**

**Description:**

* Handling File Uploads in Express.js: To handle file uploads in Express.js, we can use the multer middleware. Multer is a popular package that simplifies file upload handling. It allows us to specify the destination folder for uploaded files and other options.
* Creating an Upload Route: Now that we have the multer middleware set up, we can create a route to handle file uploads.
* Multer allows us to easily parse incoming file uploads and store them on the server.

**Source Code & Output:**



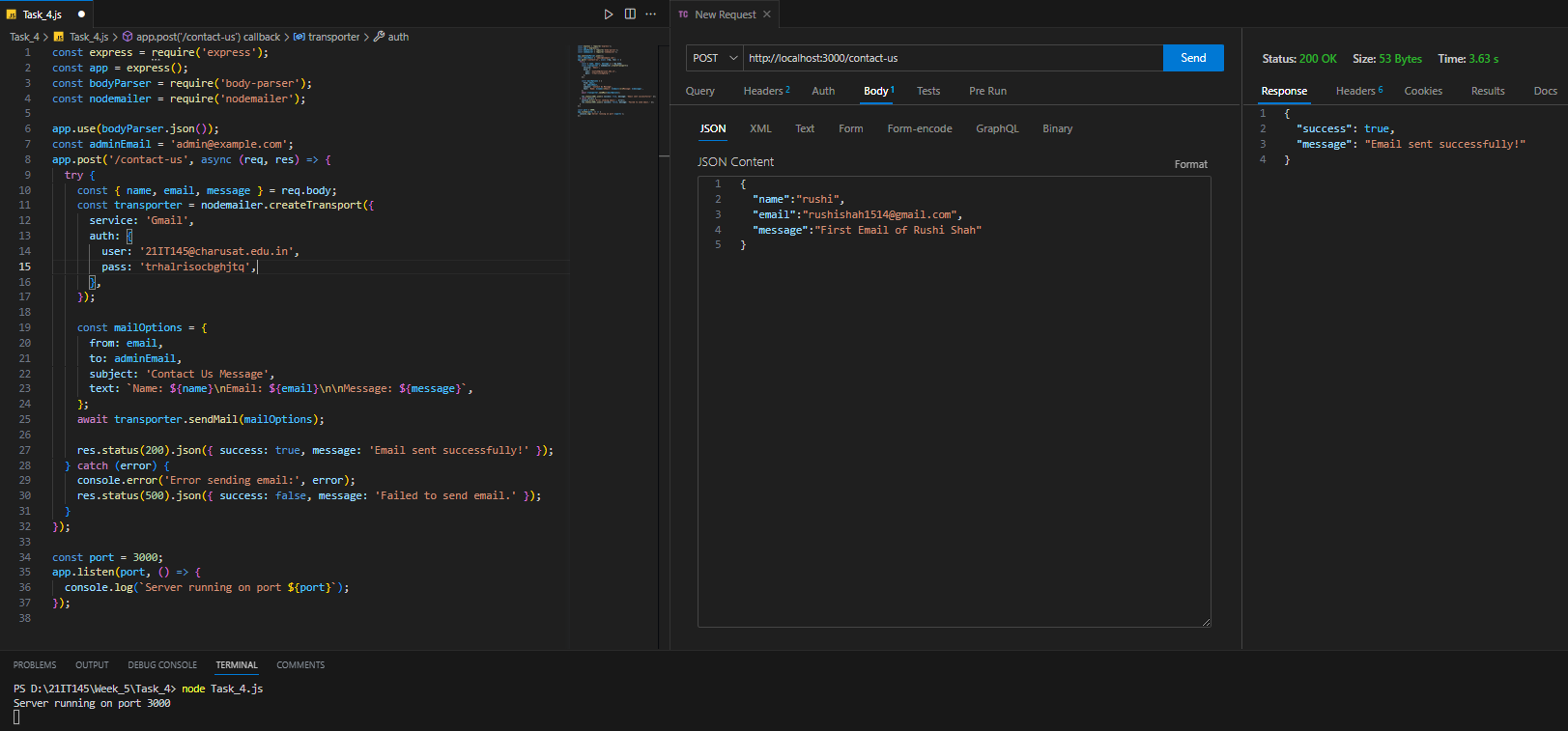
**Task-4**

**Aim:** **Create Contact us API to send email to predefined admin using nodemailer module**

**Description:**

* Nodemailer is a module for Node.js that simplifies sending emails. It supports multiple transport methods like SMTP, sendmail, and direct transport, making it versatile for different email providers.
* The Contact Us API is an endpoint in your Node.js application that allows users to send messages to predefined administrators or support teams via email.
* Provide the necessary configuration for your email service. For example, if you're using a Gmail account to send emails, you would need to provide your Gmail credentials.

**Source Code & Output:**

****

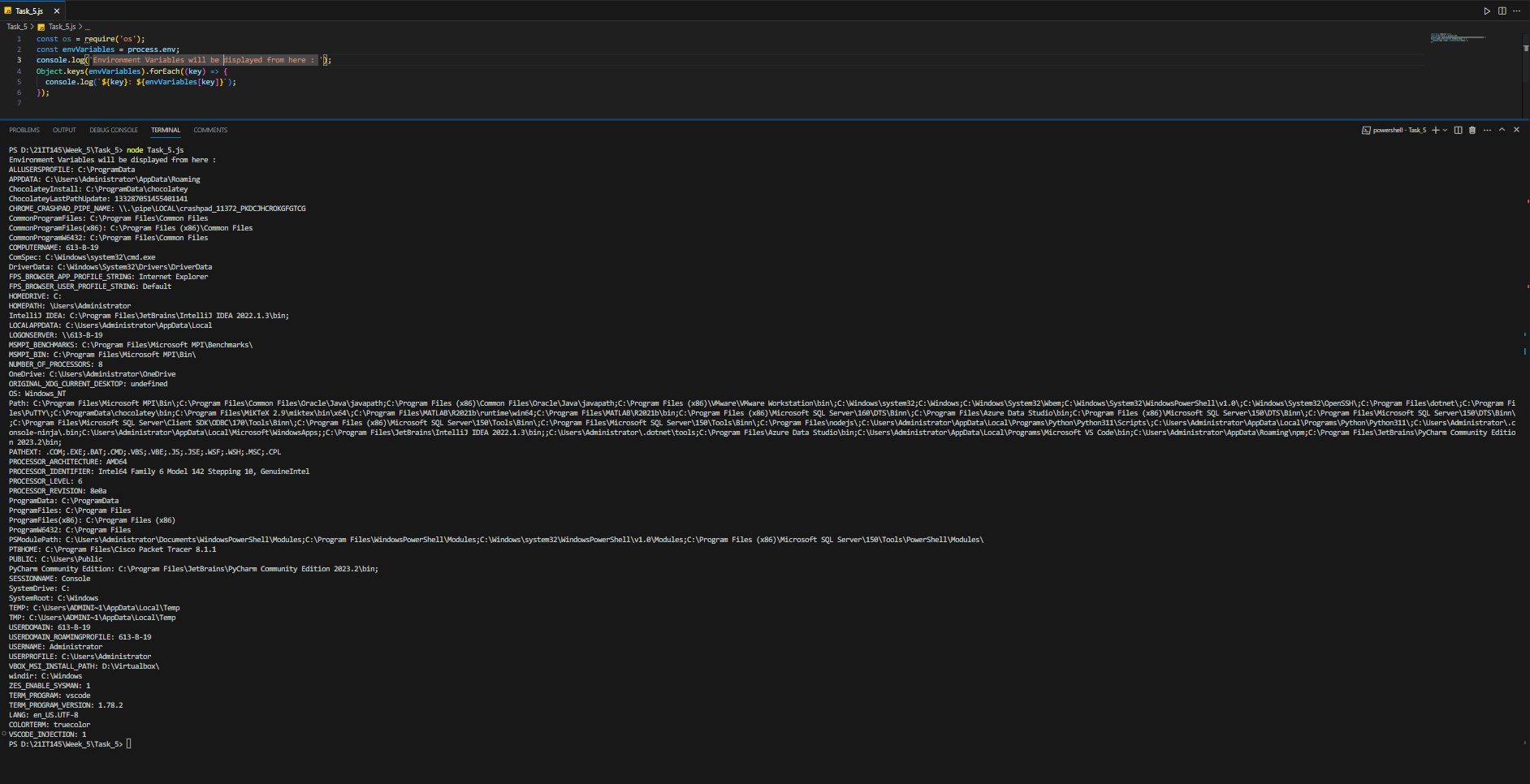
**Task-5**

**Aim: Create a program that uses the os module to display all the environment variables on the system.**

**Description:**

* Environmental variables, often referred to as environment variables or env variables, are values that can be accessed by applications at runtime to determine the environment they are running in and to configure various aspects of the application's behavior.
* In Node.js, you can use environmental variables to manage configuration settings, sensitive information, API keys, database connection strings, and more without hardcoding them directly into your code.
* Environment variables can be set in various ways depending on the platform or runtime environment in which the Node.js application is running
* The process.env object is a part of the process global object available in every Node.js application.

**Source Code & Output:**

****

**Task-6**

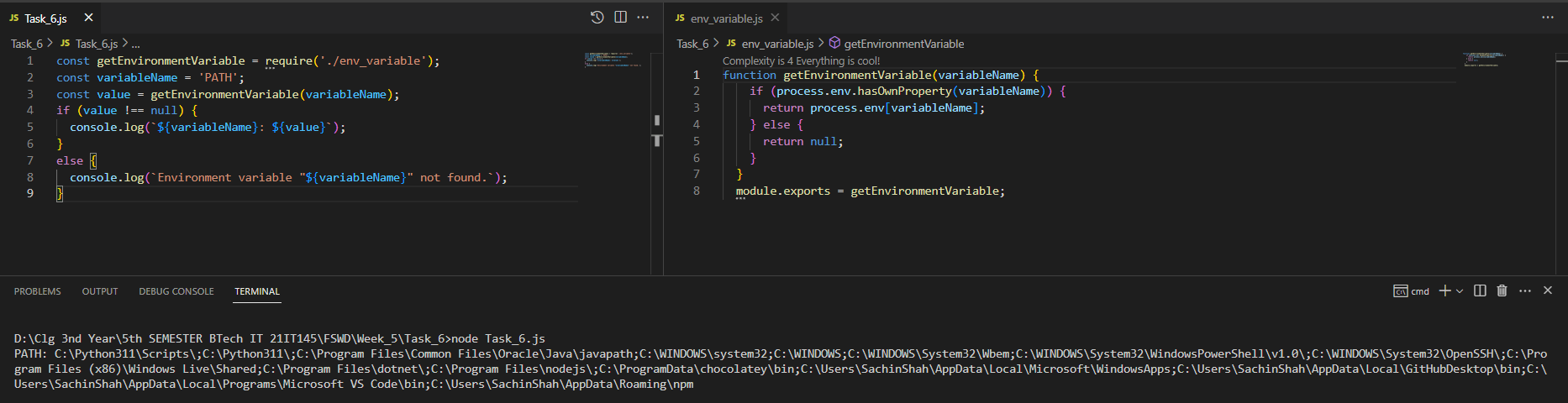
**Aim: Implement a function that takes an environment variable name as input and uses the**

**process.env object to display its corresponding value.**

**Description:**

* To use this script, ensure that you have a custom getEnvironmentVariable function defined in the env\_variable.js file.
* The function should take the variableName as an argument and return the value of the specified environment variable. Also make sure the script is executed in an environment that contains the specified environment variable.
* Keep in mind that the provided code is just a part of the implementation, and the behavior of the getEnvironmentVariable function will determine the actual functionality of the script.
* Use environment variables for more complex configuration settings and access them throughout the application as needed.

**Source Code & Output:**

****

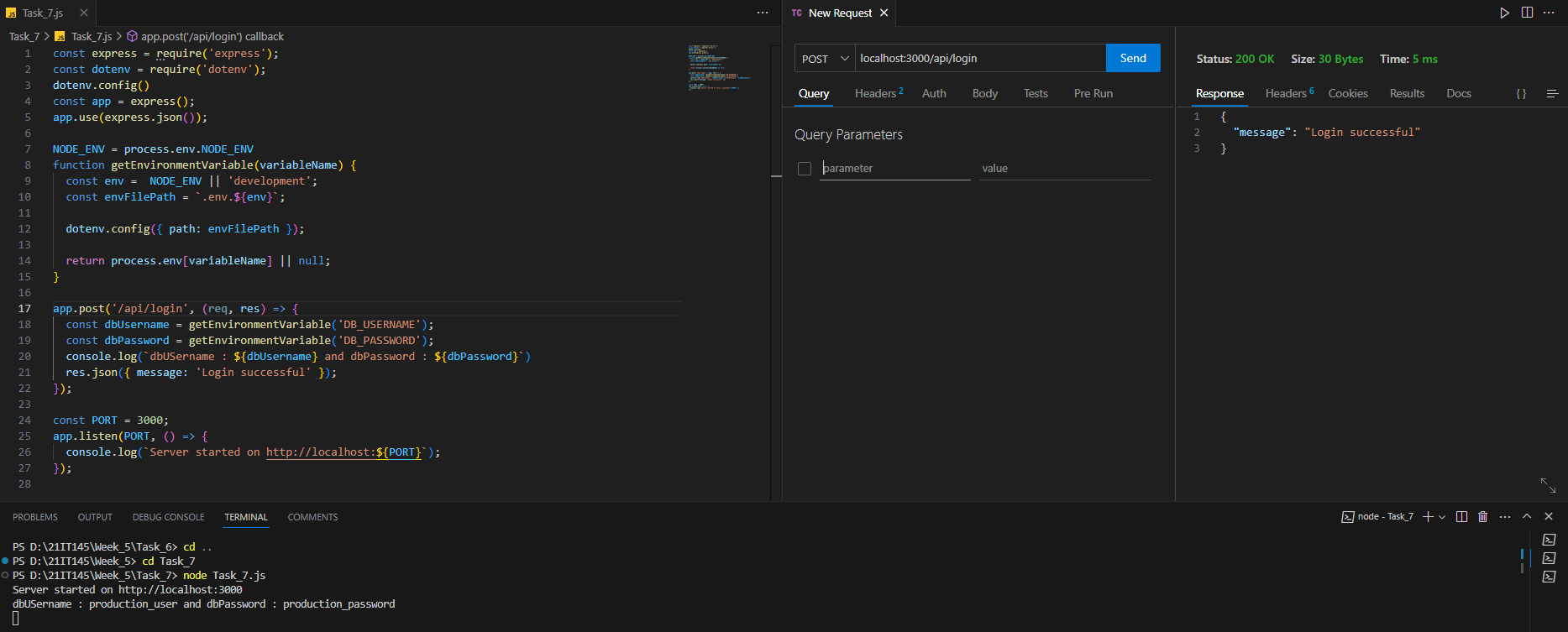
**Task-7**

**Aim: Experiments with dotenv external module & set diff. Credentials for testing, UAT and production environments.**

**Description:**

* The dotenv external module allows developers to load environment variables from .env files. These files are used to store environment-specific configuration settings, sensitive information, and credentials. By creating separate .env files for testing, UAT, and production environments, developers can ensure that the correct credentials and settings are used for each stage of the development lifecycle.
* The dotenv module simplifies the process of loading these environment-specific variables into the application, ensuring that the correct credentials are used based on the deployment environment. This approach enhances security by preventing sensitive information from being hard-coded into the codebase.

**Source Code & Output:**

****

**Learning Outcome:**

From this practical, I learnt about

* CRUD operations
* validator/joi module
* nodemailer module
* environmental variables
* .env and dotenv module

CO4: Demonstrate the use of JavaScript to fulfill the essentials of front-end development To back-end development.