



Name: Rishabh Singh

Asu id: 1224933294

Assignment: Module 1 Assignment: Web Basics

Date: 8/26/2023

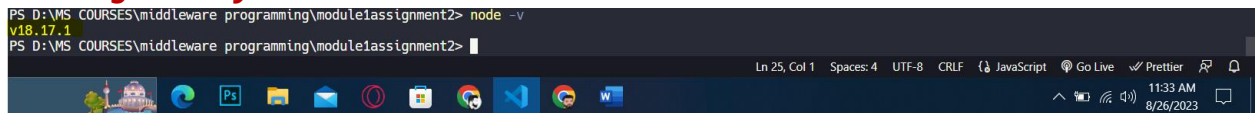
- **A summary of your understanding of Node.js**

I've had the opportunity to work with Node.js to create a web application that utilizes the Express framework and the EJS template engine. Through this experience, I've successfully built a form that collects student information from users. One crucial aspect of my work was setting up an Express application, defining routes to handle different parts of the application, and incorporating middleware like body-parser to manage form data effectively.

In my coding journey, I've gained familiarity with EJS templates, allowing me to dynamically render content on web pages. This involves embedding JavaScript within HTML, enabling me to display data seamlessly. I found this approach quite effective in creating interactive user interfaces.

Furthermore, I've developed the skill to manage form submissions. This includes processing user inputs on the server side, extracting relevant data from form fields, and incorporating this information into the application's logic. I found it intriguing how I could display the submitted data on the same page, demonstrating my ability to create dynamic views that adapt to user actions.

- **A screenshot showing the successful installation of Node.js on your local machine.**

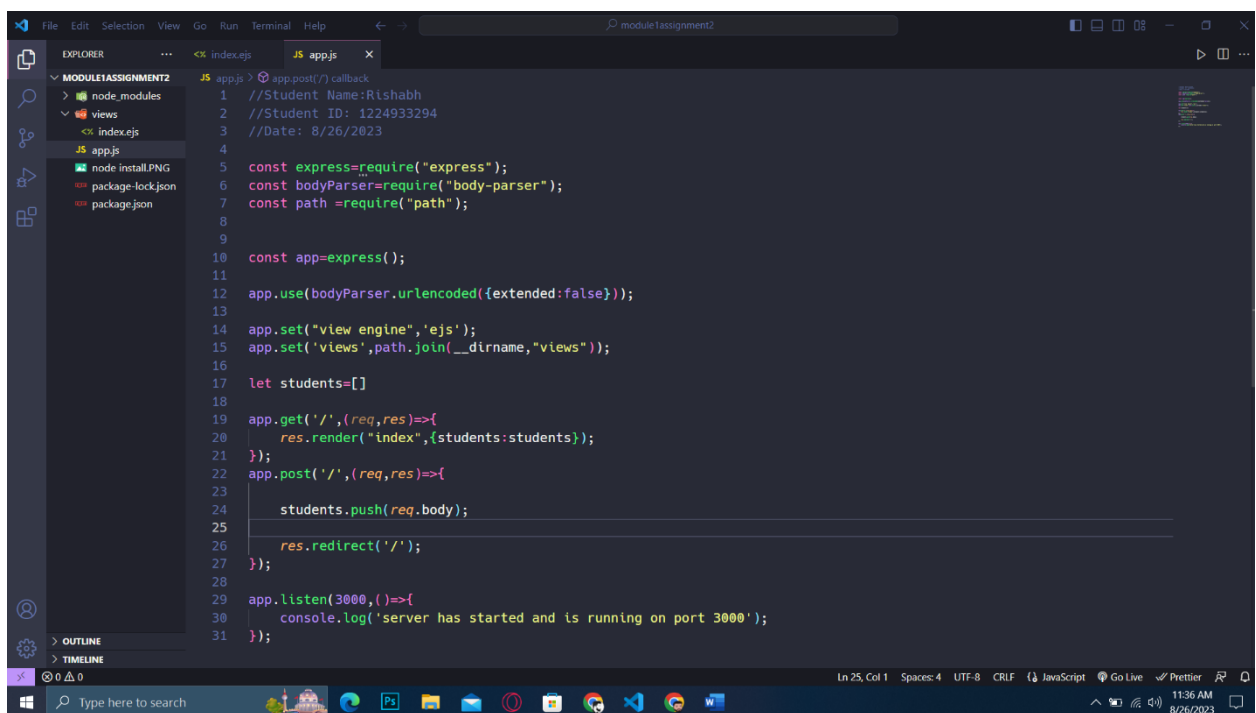
A screenshot of a Windows command prompt window. The title bar reads "COURSES\middleware programming\module1assignment2". The command prompt shows the command `node -v` being executed, and the output is `v18.17.1`. The Windows taskbar is visible at the bottom with various application icons. The system tray in the bottom right corner shows the time as 11:33 AM on 8/26/2023.

- **A brief explanation of the additional functionalities Node.js offers over standard browser-side JavaScript.**

Node.js goes beyond regular browser-side JavaScript by extending JavaScript's capabilities to the server side. It provides the ability to build backend applications, handle network requests, and manage databases using JavaScript. This enables the creation

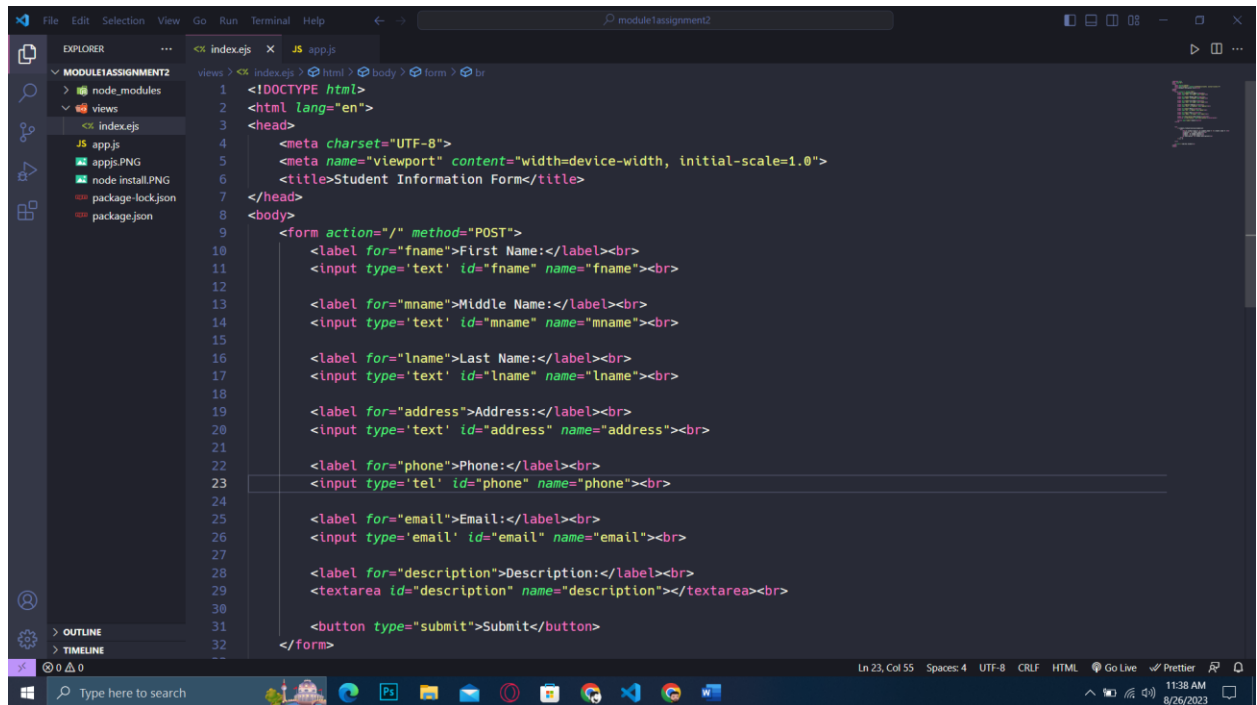
of dynamic web servers, real-time applications, and APIs. Unlike browser-based JavaScript, Node.js operates outside the browser environment, utilizing the V8 engine for fast execution. It offers features like non-blocking I/O, event-driven architecture, and access to the file system. This makes Node.js an efficient and versatile choice for building server-side applications.

- The Node.js script you created, along with a description of what it does and a screenshot of the script's output.

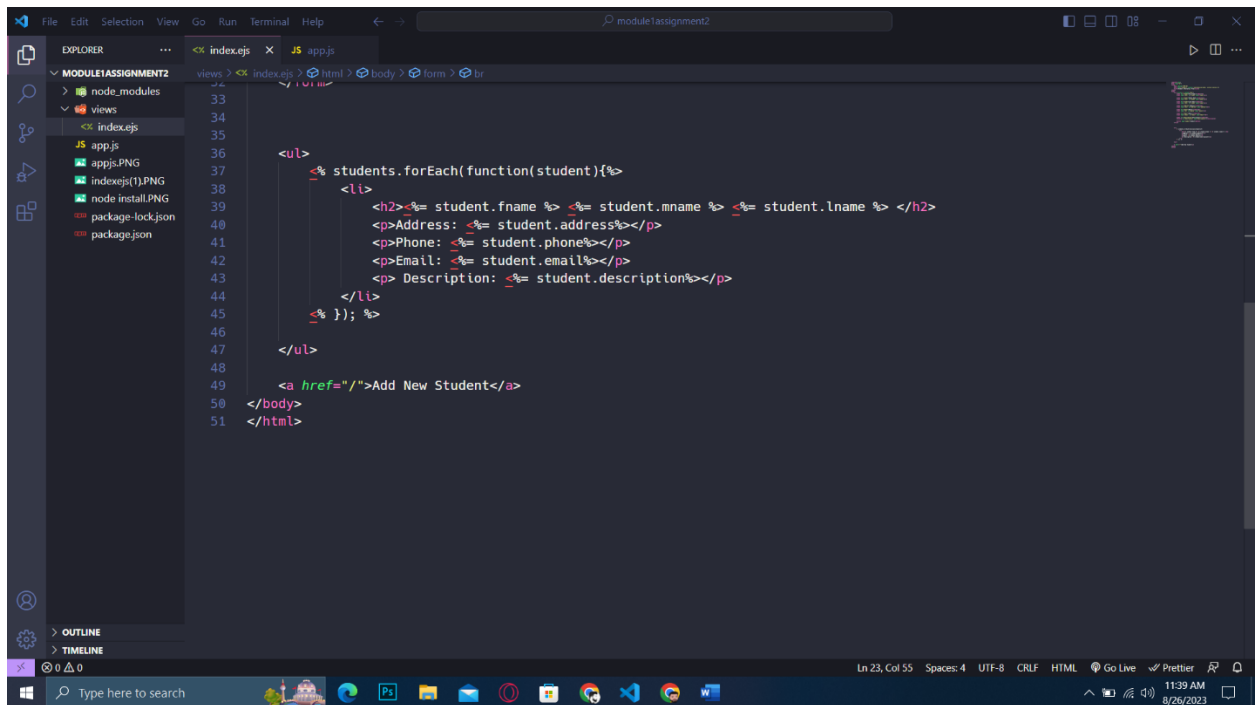
A screenshot of a Visual Studio Code editor window. The Explorer sidebar on the left shows a project named 'MODULE1ASSIGNMENT2' with a 'views' directory containing 'index.ejs' and a root directory containing 'app.js', 'node_modules', 'package-lock.json', and 'package.json'. The main editor area displays the content of 'app.js'. The code is a JavaScript script that sets up an Express web server. It includes comments for student information, requires the 'express', 'body-parser', and 'path' modules, creates an Express app, configures the view engine to 'ejs' and the view directory to 'views', initializes an empty 'students' array, defines GET and POST routes, and listens on port 3000. The status bar at the bottom indicates 'Ln 25, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'JavaScript', 'Go Live', 'Prettier', and the system time '11:36 AM 8/26/2023'.

- **Objective:** This code sets up a basic Express web server that manages student information.
- **Dependencies:** It requires express, body-parser, and path modules for server and form handling.
- **Server Setup:** An Express app is created and configured.
- **Form Handling:** It uses body-parser middleware to parse form data.
- **Template Engine:** The view engine is set to ejs, allowing dynamic HTML rendering.
- **View Location:** EJS views are stored in a "views" directory.
- **Student Data:** An empty array called students holds student information.

- **GET Route:** When a user accesses the root ("/") route, it renders the "index" view, passing the students data.
- **POST Route:** When a form is submitted, student data is added to the students array, and the page is redirected to root.
- **Server Start:** The server listens on port 3000, displaying a console message on startup.
- Overall, this code establishes a web app that accepts and displays student information using a form and Express framework.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Student Information Form</title>
7 </head>
8 <body>
9   <form action="/" method="POST">
10    <label for="fname">First Name:</label><br>
11    <input type="text" id="fname" name="fname"><br>
12
13    <label for="mname">Middle Name:</label><br>
14    <input type="text" id="mname" name="mname"><br>
15
16    <label for="lname">Last Name:</label><br>
17    <input type="text" id="lname" name="lname"><br>
18
19    <label for="address">Address:</label><br>
20    <input type="text" id="address" name="address"><br>
21
22    <label for="phone">Phone:</label><br>
23    <input type="tel" id="phone" name="phone"><br>
24
25    <label for="email">Email:</label><br>
26    <input type="email" id="email" name="email"><br>
27
28    <label for="description">Description:</label><br>
29    <textarea id="description" name="description"></textarea><br>
30
31    <button type="submit">Submit</button>
32  </form>
```



The screenshot shows a VS Code editor window with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'module1assignment2' with files like 'index.ejs', 'app.js', 'package-lock.json', and 'package.json'. The code editor displays the content of 'index.ejs', which is an EJS template. The template includes a form for adding a new student and a list to display existing students. The form has input fields for first name, middle name, last name, address, phone, email, and a description, along with a 'Submit' button. The list displays student information using EJS tags to inject data from a 'students' array. The status bar at the bottom indicates the current line and column (Ln 23, Col 55) and shows various icons for file operations and settings.

```
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

<ul>
  <li>
    <h2><%= student.fname %> <%= student.mname %> <%= student.lname %> </h2>
    <p>Address: <%= student.address%></p>
    <p>Phone: <%= student.phone%></p>
    <p>Email: <%= student.email%></p>
    <p>Description: <%= student.description%></p>
  </li>
</ul>

<%= }); %>

<a href="/">Add New Student</a>
</body>
</html>
```

- **Form Setup:** This HTML code creates a form to collect student information.
- **Form Fields:** It includes input fields for first name, middle name, last name, address, phone, email, and a description.
- **Submit Button:** A "Submit" button triggers the form submission.
- **Server Interaction:** The form action is set to "/" (root), using the POST method to send data to the server.
- **Data Display:** Below the form, a list displays student data using an EJS loop.
- **Dynamic Rendering:** EJS tags like `<%= student.fname %>` dynamically inject data into the HTML.
- **User Interaction:** An "Add New Student" link lets users return to the form.
- **Overall Purpose:** This HTML code creates an interactive web page to input, display, and manage student information using a form and dynamic rendering.

OUTPUT:

Module 1 Assignment 2: Basic D: x Student Information Form x +

localhost:3000

First Name:

Middle Name:

Last Name:

Address:

Phone:

Email:

Description:

• **Rishabh Singh**

Address: 1701 E 8th st,
Phone: +16025825903
Email: rvsing159@asu.edu
Description: Hi, Dev This side <3

[Add New Student](#)

Type here to search

11:40 AM
8/26/2023

- **Your reflection on the experience of setting up and working with Node.js.**

Working with Node.js has been an enlightening experience for me. It's impressive how Node.js extends JavaScript beyond the browser and empowers me to build powerful server-side

applications. The ability to utilize the same language on both the client and server sides brings a sense of cohesion to my development process. Setting up an Express web server felt intuitive, and the integration of middleware like body-parser streamlined form handling. Incorporating the EJS template engine allowed me to craft dynamic views effortlessly. What's particularly intriguing is the event-driven and non-blocking architecture that enhances the application's efficiency. This experience has deepened my understanding of full-stack development and reinforced the versatility of JavaScript in driving both front-end and back-end functionalities.