**EXPERIMENT 08**

**Aim:** Build a Note Taking Application using Node.js

**Lab Objective:** Students will be able to:

 Understand and implement Node.js to make Note Taking Application

**Theory:**

Node.js is an open-source, server-side runtime environment that is built on the Chrome V8 JavaScript engine. It allows developers to write server-side code using JavaScript, making it a popular choice for building web applications and APIs. Node.js provides an event-driven, non-blocking I/O model, which makes it efficient and well-suited for handling concurrent connections.

Key aspects of Node.js:

* JavaScript on the Server: Node.js allows you to use JavaScript on the server-side. This means you can use the same language (JavaScript) on both the client and server sides of your application, which can simplify development.
* Asynchronous and Non-blocking: Node.js is designed to be non-blocking and event-driven. It uses callback functions and event loops to handle multiple requests simultaneously without blocking the execution of other code. This is particularly useful for handling I/O operations, such as reading files, making network requests, and interacting with databases.
* NPM (Node Package Manager): Node.js comes with npm, a package manager that makes it easy to install, manage, and share packages (libraries) created by other developers. NPM is a vast ecosystem with thousands of packages that you can use in your Node.js applications.
* V8 JavaScript Engine: Node.js uses the V8 JavaScript engine, developed by Google, which is known for its speed and performance. This makes Node.js efficient in executing JavaScript code.
* Event-Driven Architecture: Node.js is built around an event-driven architecture. It provides a set of APIs for handling events, allowing you to create applications that respond to various events, such as incoming HTTP requests or file system changes.

Node.js is a runtime environment for executing JavaScript on the server-side. Here's how to implement it:

1. Install Node.js: Download and install Node.js from the official website.
2. Create a Project Directory: Make a directory for your project using mkdir and navigate to it using cd.
3. Initialize a Node.js Project: Create a package.json file using npm init to manage project information and dependencies.
4. Write JavaScript Code: Create JavaScript files for your project in the project directory.
5. Install Dependencies: Use npm install to add external packages or libraries if needed.
6. Create a Node.js Application: Write Node.js code to build your application, which can be a web server, API, or any other server-side application.
7. Run the Application: Execute your Node.js application with node filename.js.
8. Test Your Application: Test your application in a web browser or with tools like curl or Postman.
9. Deploy Your Application: Host your Node.js app on a server or cloud platform, such as Heroku, AWS, or Azure.

Maintain and Improve: Regularly maintain and update your application, monitoring performance and security, and adding new features as needed. Node.js is versatile and applicable to various types of projects.

**Source code:**

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Note Taking App</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="container">

<h1>My Notes</h1>

<div class="note-form">

<input type="text" id="note-title" placeholder="Title">

<textarea id="note-body" placeholder="Take a note..."></textarea>

<button id="add-note">Add Note</button>

</div>

<div id="notes-list">

<!-- Notes will be displayed here -->

</div>

</div>

<script src="script.js"></script>

</body>

</html>

**CSS:**

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

margin: 0;

padding: 0;

}

.container {

max-width: 600px;

margin: 0 auto;

background-color: #fff;

padding: 20px;

border-radius: 5px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

}

h1 {

text-align: center;

}

.note-form {

margin-top: 20px;

}

input[type="text"],

textarea {

width: 100%;

padding: 10px;

margin: 5px 0;

border: 1px solid #ccc;

border-radius: 5px;

}

button {

background-color: #007BFF;

color: #fff;

border: none;

border-radius: 5px;

padding: 10px 20px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

#notes-list {

margin-top: 20px;

}

.note {

background-color: #fff;

border: 1px solid #ccc;

border-radius: 5px;

padding: 10px;

margin-bottom: 10px;

}

**JS:**

document.addEventListener('DOMContentLoaded', () => {

const noteTitleInput = document.getElementById('note-title');

const noteBodyInput = document.getElementById('note-body');

const addNoteButton = document.getElementById('add-note');

const notesList = document.getElementById('notes-list');

// Event listener for adding a new note

addNoteButton.addEventListener('click', () => {

const title = noteTitleInput.value;

const body = noteBodyInput.value;

if (title && body) {

const note = document.createElement('div');

note.classList.add('note');

note.innerHTML = `

<h3>${title}</h3>

<p>${body}</p>

`;

notesList.appendChild(note);

// You can also send the note to the Node.js server for storage here

// You'll need to make an HTTP POST request to the server

// Clear the input fields

noteTitleInput.value = '';

noteBodyInput.value = '';

}

});

});

**Node.js:**

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());

const notes = [];

// Get all notes

app.get('/api/notes', (req, res) => {

res.json(notes);

});

// Create a new note

app.post('/api/notes', (req, res) => {

const newNote = req.body;

notes.push(newNote);

res.json(newNote);

});

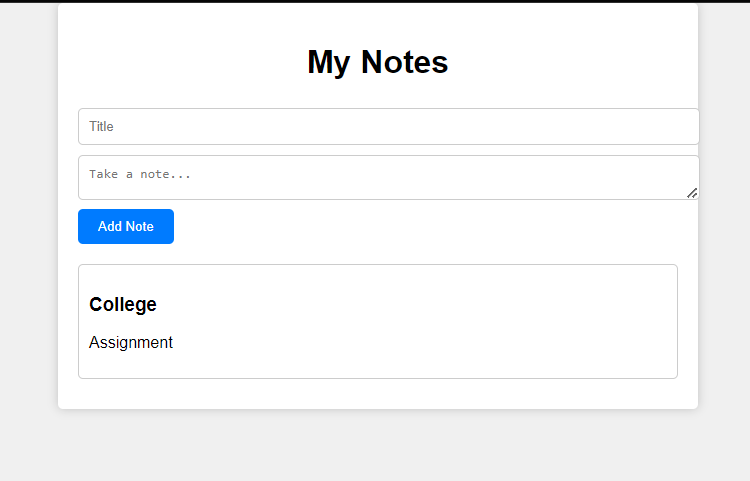
const port = process.env.PORT || 3000;

app.listen(port, () => {

console.log(`Server is running on port ${port}`);

});

**Output (Web Page):**



**Lab Outcome**: Students were able to:

Implement concepts of Node.js and nuild a note taking app.

**Conclusion:**

**COs attained:**

CO1 (Implement interactive web page(s) using HTML)

CO2 (Create Responsive Web Design with CSS & Bootstrap)

CO3 (Design and develop web applications using JavaScript)

**POs attained:**

PO 1: ENGINEERING KNOWLEDGE (Apply Knowledge of Mathematics, Science,

engineering fundamentals and an engineering specialization to the solution of complex engineering problems.)

PO 2: PROBLEM ANALYSIS (Identify, formulate, research literature and analyse complex

engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.)

PO 3: DESIGN / DEVELOPMENT OF SOLUTIONS (Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.)

PO 5:MODERN TOOL USAGE (Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.)

**PEOs achieved:**

PEO 1: To prepare learners with a strong foundation in the area of Information Technology required solving real life problems arising from software technology. (Knowledge)(CURRICULAR)

PEO 3: To prepare learners to understand the need for lifelong learning with effective written

and oral communication skills and to be able to readily adapt to new software engineering environments. (PRESENTATION AND GROWTH) **.**