awt progress # awt01 # Web Technology Lab – Experiments Summary

This repository contains all experiments and classwork done as part of the Advanced Web Technology (AWT) lab.
Each experiment focuses on different web technologies including HTML, CSS, JavaScript, jQuery, AngularJS, and Node.js.

Table of Contents

Experiment 1 – HTML & CSS Basics

What I Learned

- Creating web pages using *HTML elements* and *semantic tags*
- Styling pages with *CSS selectors, classes, and IDs*
- Building responsive layouts using *flexbox* and *media queries*

code #### index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>LAB EXP 1</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.mi
n.js"></script>
    <link rel="stylesheet" href="style.css">
</head>
<body>

    <div class="container">
        <h1>1. Right-Click Disabled</h1>
        <p>Try right-clicking anywhere on this page. The context menu will no
```

```
t appear.</p>
    </div>

    <hr>

    <div class="container">
        <h1>2. Show/Hide Message</h1>
        <button id="showBtn">Show Message</button>
        <button id="hideBtn">Hide Message</button>
        <div id="messageDiv">
            <p>Hello! This message can be hidden and shown using the buttons
above.</p>
        </div>
    </div>

    <hr>

    <div class="container">
        <h1>3. Paragraph Color Change on Hover</h1>
        <p class="hover-para">This paragraph changes color.</p>
    </div>

    <hr>

    <div class="container" style="height: 1200px;">
        <h1>4. Scroll to Top</h1>
        <p>Scroll down this page to see the image in the bottom-right corner.
 Clicking it will bring you back to the top of the page.</p>
    </div>


    <img src="C:\Users\hp\Pictures\Screenshots\Screenshot 2025-08-25 231221.p
ng" alt="Scroll to Top" id="scrollToTopBtn">

    <script src="exp1.js"></script>

</body>
</html>
```

style.css

```
/* General Page Styling */
body {
  font-family: 'Segoe UI', Arial, sans-serif;
  background-color: #fafafa;
  color: #333;
  margin: 30px auto;
  max-width: 800px;
  line-height: 1.6;
```

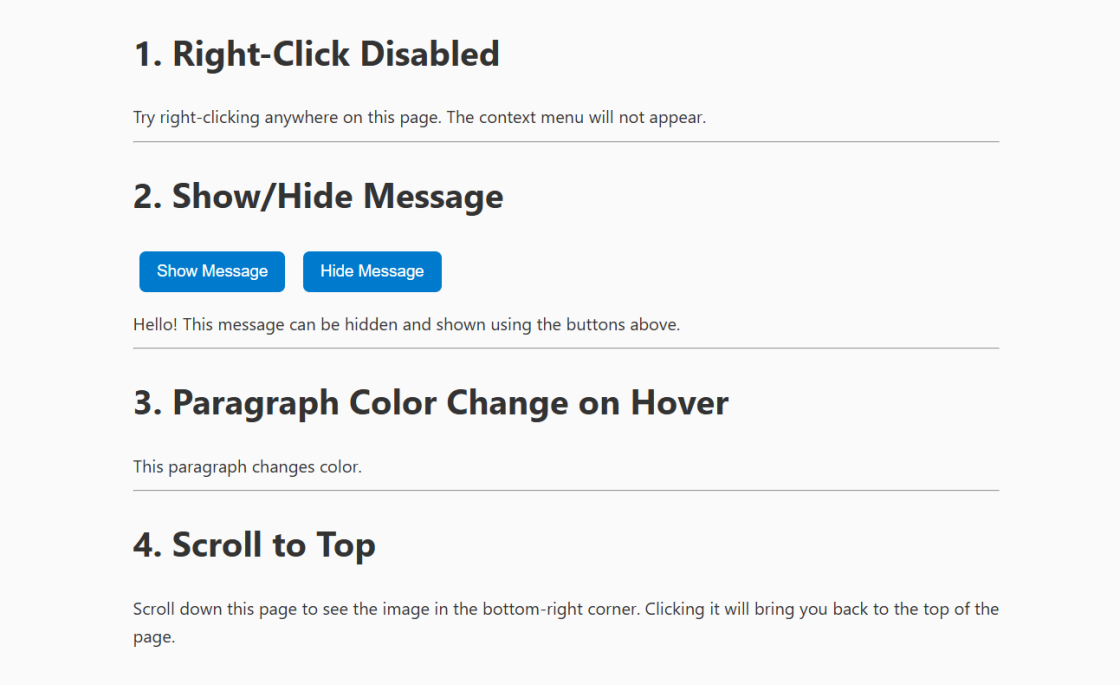
```css
}

/* Headings */
h2 {
  text-align: center;
  color: #007acc;
  margin-bottom: 20px;
}

h3 {
  color: #333;
  border-left: 5px solid #007acc;
  padding-left: 10px;
  margin-top: 30px;
}

/* Scroll-to-top Image */
#topImage {
  width: 60px;
  cursor: pointer;
  margin: 10px 0;
  transition: transform 0.3s ease;
}
#topImage:hover {
  transform: scale(1.2);
}

/* Paragraph Styling */
p {
  margin: 10px 0;
}
.hoverText {
  transition: color 0.3s ease;
  cursor: pointer;
}

/* Buttons */
button {
  background-color: #007acc;
  color: white;
  border: none;
  border-radius: 6px;
  padding: 10px 16px;
  margin: 6px;
  font-size: 15px;
  cursor: pointer;
  transition: background-color 0.3s ease, transform 0.2s ease;
}
button:hover {
```

```
  background-color: #005f99;
  transform: scale(1.05);
}

/* Message Box */
#messageBox {
  display: none;
  padding: 15px;
  margin-top: 10px;
  border
}
```

exp1.js

```
$(document).on("selectstart", function(e){
    e.preventDefault();
});

$(document).on("keydown", function(e) {
    if (e.ctrlKey && e.keyCode === 67) e.preventDefault();
});

 $(document).ready(function() {
            // 1. Disable the right-click menu
            $(document).on("contextmenu", function(e) {
                e.preventDefault();
            });

            // 2. Display and hide a message
            $("#showBtn").click(function() {
                $("#messageDiv").show('slow');
            });
            $("#hideBtn").click(function() {
                $("#messageDiv").hide('slow');
            });

            // 3. Change paragraph color on mouseover
            $(".hover-para").mouseover(function() {
                $(this).css("color", "red");
            });
            $(".hover-para").mouseout(function() {
                $(this).css("color", "#333");
            });

            // 4. Click an image to scroll to the top
            $("#scrollToTopBtn").click(function() {
                $("html, body").animate({ scrollTop: 0 }, 'slow');
            })

        });
```

output



Output Screenshot

Challenges Faced

- Aligning elements correctly across different screen sizes
- Understanding the difference between *inline, **inline-block, and** block* elements

---

Experiment 2 – JavaScript Fundamentals

What I Learned

- Writing *JavaScript functions* and handling *events*
- Working with *loops, conditions, and arrays*
- Using the *DOM API* to change webpage content dynamically

code ####index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Experiment 2 - jQuery CSS and Events</title>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

```
  <link rel="stylesheet" href="exp2.css">
</head>
<body>

  <h1>Experiment 2: jQuery CSS and Events</h1>

  <h3>1. Add a class to an element</h3>
  <p id="text">Click the button to highlight this text!</p>
  <button id="addClassBtn">Add Class</button>

  <h3>2. Access the position of an element</h3>
  <p id="positionText">Click to see this element's position!</p>
  <button id="positionBtn">Show Position</button>

  <h3>3. Animate multiple CSS properties</h3>
  <div id="animateBox"></div>
  <button id="animateBtn">Animate Box</button>

  <script src="exp2.js"></script>
</body>
</html>
```

exp2.css

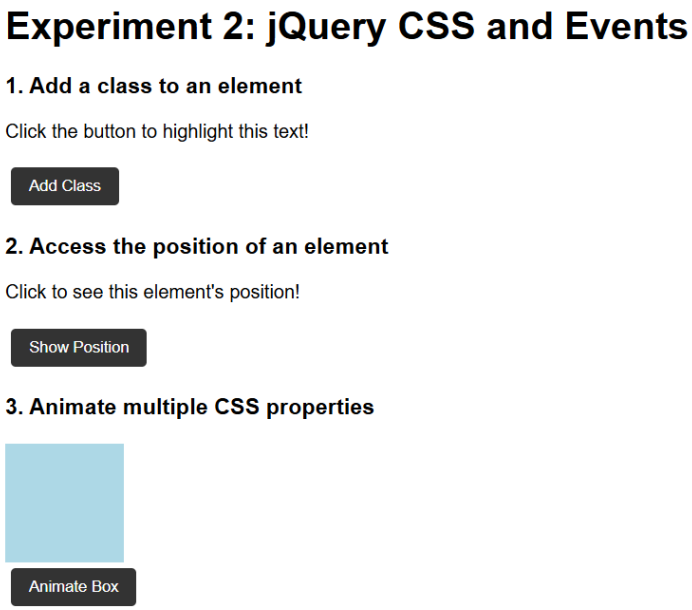
```
body {
  font-family: Arial, sans-serif;
  padding: 20px;
}

.highlight {
  background: red;
  font-weight: bold;
  padding: 10px;
}

#animateBox {
  width: 100px;
  height: 100px;
  background: lightblue;
  position: relative;
  margin-top: 20px;
}

button {
  margin: 5px;
  padding: 8px 15px;
  border: none;
  background: #333;
  color: white;
  border-radius: 4px;
```

```
  cursor: pointer;
}

button:hover {
  background: #555;
}
```

exp2.js

```
$(document).ready(function () {
  // 1. Add a class
  $("#addClassBtn").click(function () {
    $("#text").addClass("highlight");
  });

  // 2. Show position of element
  $("#positionBtn").click(function () {
    let pos = $("#positionText").position();
    alert("Top: " + pos.top + ", Left: " + pos.left);
  });

  // 3. Animate multiple CSS properties
  $("#animateBtn").click(function () {
    $("#animateBox").animate({
      left: "+=100px",
      top: "+=50px",
      width: "150px",
      height: "150px",
      opacity: 0.7
    }, 1000);
  });
});
```

output



Output Screenshot

Challenges Faced

- Debugging DOM errors in the browser console
- Managing variable scopes and timing issues with event listeners

---

Experiment 3 – jQuery and DOM Manipulation

What I Learned

- Simplifying DOM manipulation with *jQuery selectors*
- Handling *click* and *hover* events easily
- Implementing *form validation* and effects like fade/slide

code #### index.html

```
<!doctype html>
<html ng-app="tableApp">
<head>
  <meta charset="utf-8">
  <title>Experiment 3 - AngularJS Tables</title>
```

```
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.
min.js"></script>
  <script src="exp3.js"></script>
  <style>
    table { border-collapse: collapse; width: 100%; }
    th, td { border: 1px solid #ccc; padding: 8px; }
    tr.even { background: #f8f8f8; }
    tr.odd { background: #ffffff; }
    th { background: #eee; }
  </style>
</head>
<body ng-controller="TableController as ctrl">
  <div class="container">
    <h1>AngularJS Table Examples</h1>

    <section>
      <h2>1. Display a Table</h2>
      <table>
        <thead>
          <tr>
            <th>#</th>
            <th>Name</th>
            <th>Age</th>
            <th>Department</th>
          </tr>
        </thead>
        <tbody>
          <tr ng-repeat="student in ctrl.students track by $index" ng-class-o
dd="'odd'" ng-class-even="'even'">
            <td>{{$index + 1}}</td>
            <td>{{student.name}}</td>
            <td>{{student.age}}</td>
            <td>{{student.dept}}</td>
          </tr>
        </tbody>
      </table>
    </section>

    <section>
      <h2>2. Display contents with orderBy filter</h2>
      <label>Sort by:
        <select ng-model="ctrl.sortKey">
          <option value="name">Name</option>
          <option value="age">Age</option>
          <option value="dept">Department</option>
        </select>
        <label><input type="checkbox" ng-model="ctrl.reverse"> Reverse</label>
      </label>
```

```
      <table>
        <thead>
          <tr>
            <th>#</th>
            <th>Name</th>
            <th>Age</th>
            <th>Department</th>
          </tr>
        </thead>
        <tbody>
          <tr ng-repeat="student in ctrl.students | orderBy:ctrl.sortKey:ctrl.
reverse track by $index" ng-class-even="'even'" ng-class-odd="'odd'">
            <td>{{$index + 1}}</td>
            <td>{{student.name}}</td>
            <td>{{student.age}}</td>
            <td>{{student.dept}}</td>
          </tr>
        </tbody>
      </table>
    </section>

    <section>
      <h2>3. Display Table with even and odd rows (styling already shown)</h2>
      <p>Notice rows have alternating backgrounds using <code>ng-class-even</
code> and <code>ng-class-odd</code>.</p>
    </section>
  </div>
</body>
</html>
```

exp3.js

```
angular.module('tableApp', [])
  .controller('TableController', function() {
    const vm = this;

    vm.students = [
      { name: 'Asha', age: 22, dept: 'CSE' },
      { name: 'Bikram', age: 24, dept: 'ECE' },
      { name: 'Charu', age: 21, dept: 'ME' },
      { name: 'Deep', age: 23, dept: 'CSE' },
      { name: 'Esha', age: 20, dept: 'EE' }
    ];

    vm.sortKey = 'name';
    vm.reverse = false;
  });
```

output

AngularJS Table Examples

1. Display a Table

#	Name	Age	Department
1	Asha	22	CSE
2	Bikram	24	ECE
3	Charu	21	ME
4	Deep	23	CSE
5	Esha	20	EE

2. Display contents with orderBy filter

Sort by: Name ☐ Reverse

#	Name	Age	Department
1	Asha	22	CSE
2	Bikram	24	ECE
3	Charu	21	ME
4	Deep	23	CSE
5	Esha	20	EE

3. Display Table with even and odd rows (styling already shown)

Notice rows have alternating backgrounds using `ng-class-even` and `ng-class-odd`.

Output Screenshot

Challenges Faced

- Remembering the correct jQuery syntax
- Understanding the difference between **$(document).ready()** and normal JS load timing

---

Experiment 4 – AngularJS MVC Architecture

What I Learned

- Understanding *Model-View-Controller (MVC)* in AngularJS
- Using *controllers, directives, and filters*
- Implementing *two-way data binding*

code #### bill.html

```
<!DOCTYPE html>
<html lang="en" ng-app="billApp">
<head>
  <meta charset="UTF-8">
  <title>Bill Payment Record</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.
min.js"></script>
  <style>
    body { font-family: Arial, sans-serif; margin: 20px; }
    table { border-collapse: collapse; width: 100%; margin-top: 20px; }
```

```
    th, td { border: 1px solid #ccc; padding: 8px; text-align: left; }
    th { background: #eee; }
    input, button { margin: 5px; padding: 6px; }
  </style>
</head>
<body ng-controller="BillController">

  <h2>Bill Payment Record</h2>

  <!-- Form to add new records -->
  <form name="billForm" ng-submit="addRecord(billForm)" novalidate>
    <label>
      Name:
      <input type="text" name="name" ng-model="newRecord.name" required>
    </label>
    <label>
      Amount:
      <input type="number" name="amount" ng-model="newRecord.amount" required
 min="0">
    </label>
    <label>
      Date:
      <input type="date" name="date" ng-model="newRecord.date" required>
    </label>
    <button type="submit" ng-disabled="billForm.$invalid">Add</button>
  </form>

  <!-- Records table -->
  <table ng-if="records.length > 0">
    <thead>
      <tr>
        <th>#</th>
        <th>Name</th>
        <th>Amount</th>
        <th>Date</th>
      </tr>
    </thead>
    <tbody>
      <tr ng-repeat="record in records track by record.id">
        <td>{{$index + 1}}</td>
        <td>{{record.name}}</td>
        <td>{{record.amount | currency}}</td>
        <td>{{record.date | date:'mediumDate'}}</td>
      </tr>
    </tbody>
  </table>

  <script>
    angular.module('billApp', [])
```

```
      .controller('BillController', function($scope) {
        // Initial records
        $scope.records = [
          { id: 1, name: 'Electricity', amount: 1200, date: '2025-07-01' },
          { id: 2, name: 'Internet', amount: 799, date: '2025-07-05' }
        ];

        $scope.newRecord = {};

        // Add new record
        $scope.addRecord = function(form) {
          if (form.$valid) {
            let newId = $scope.records.length + 1;
            $scope.records.push({
              id: newId,
              name: $scope.newRecord.name,
              amount: $scope.newRecord.amount,
              date: $scope.newRecord.date
            });
            $scope.newRecord = {};
            form.$setPristine();
            form.$setUntouched();
          }
        };
      });
  </script>
</body>
</html>
```

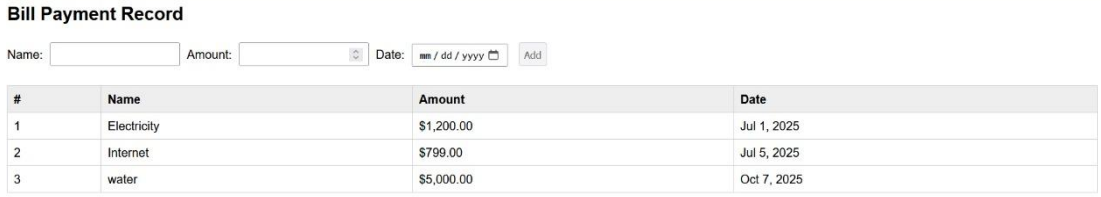
form.html

```
<!DOCTYPE html>
<html lang="en" ng-app="formApp">
<head>
  <meta charset="UTF-8">
  <title>AngularJS Registration Form</title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.3/angular.
min.js"></script>

  <style>
    .error { color: red; font-size: 14px; }
    input.ng-invalid.ng-touched { border: 2px solid red; }
    input.ng-valid.ng-touched { border: 2px solid green; }
    .success { color: green; margin-top: 10px; }
  </style>

</head>

<body ng-controller="FormController">
```

```
  <h2>User Registration</h2>
  <form name="regForm" novalidate ng-submit="register(regForm)">
    <label>Name:
      <input type="text" name="name" ng-model="user.name" required ng-minleng
th="3">
    </label>
    <div class="error"
         ng-show="(regForm.name.$touched || submitted) && regForm.name.$inval
id">
      Name must be at least 3 characters.
    </div>
    <br><br>

    <label>Email:
      <input type="email" name="email" ng-model="user.email" required>
    </label>
    <div class="error"
         ng-show="(regForm.email.$touched || submitted) && regForm.email.$inv
alid">
      Enter a valid email.
    </div>
    <br><br>

    <label>Password:
      <input type="password" name="password" ng-model="user.password" require
d ng-minlength="6">
    </label>
    <div class="error"
         ng-show="(regForm.password.$touched || submitted) && regForm.passwor
d.$invalid">
      Password must be at least 6 characters.
    </div>
    <br><br>

    <button type="submit">Register</button>
  </form>

  <p class="success" ng-if="success">{{success}}</p>

  <script>
    angular.module('formApp', [])
      .controller('FormController', function($scope) {
        $scope.user = {};
        $scope.submitted = false;
        $scope.success = '';

        $scope.register = function(form) {
          if (form.$valid) {
            // In real app, send data to server
```

```
            $scope.success = 'Registration successful for ' + $scope.user.nam
e;
            $scope.user = {};
            form.$setPristine();  // Reset form state
            form.$setUntouched(); // Reset touched state
            $scope.submitted = false;
          } else {
            $scope.success = '';
            $scope.submitted = true; // Show validation messages
          }
        };
      });
  </script>
</body>
</html>
```

output



Bill Payment Record

Name: Amount: Date: mm / dd / yyyy Add

#	Name	Amount	Date
1	Electricity	$1,200.00	Jul 1, 2025
2	Internet	$799.00	Jul 5, 2025
3	water	$5,000.00	Oct 7, 2025

User Registration

Name:

Email:

Password:

Register

Registration successful for NEHA

Challenges Faced

- Linking the AngularJS library properly
- Handling scope and data flow between controller and view

Experiment 5 & 6 – Node.js and Express Basics

What I Learned

- Creating a *simple Node.js server*
- Using *Express.js* to handle routes and responses
- Serving static files and building basic APIs

code #### server.js

```
const express = require('express');
const app = express();
const PORT = process.env.PORT || 3000;

// Hello World endpoint
app.get('/', (req, res) => {
  res.send('Hello, World!');
});

// String replacement endpoint
app.get('/replace', (req, res) => {
  const { text } = req.query;
  if (!text) {
    return res.status(400).json({ error: 'Text parameter is required' });
  }

  const regex = /a{2,}/g;
  const result = text.replace(regex, 'b');
  res.json({ original: text, replaced: result });
});

// Calculator endpoint
app.get('/calculate', (req, res) => {
  const { operation, num1, num2 } = req.query;
  const n1 = parseFloat(num1);
  const n2 = parseFloat(num2);

  if (isNaN(n1) || isNaN(n2)) {
    return res.status(400).json({ error: 'Invalid numbers provided' });
  }

  let result;
  switch (operation) {
    case 'add':
      result = n1 + n2;
```

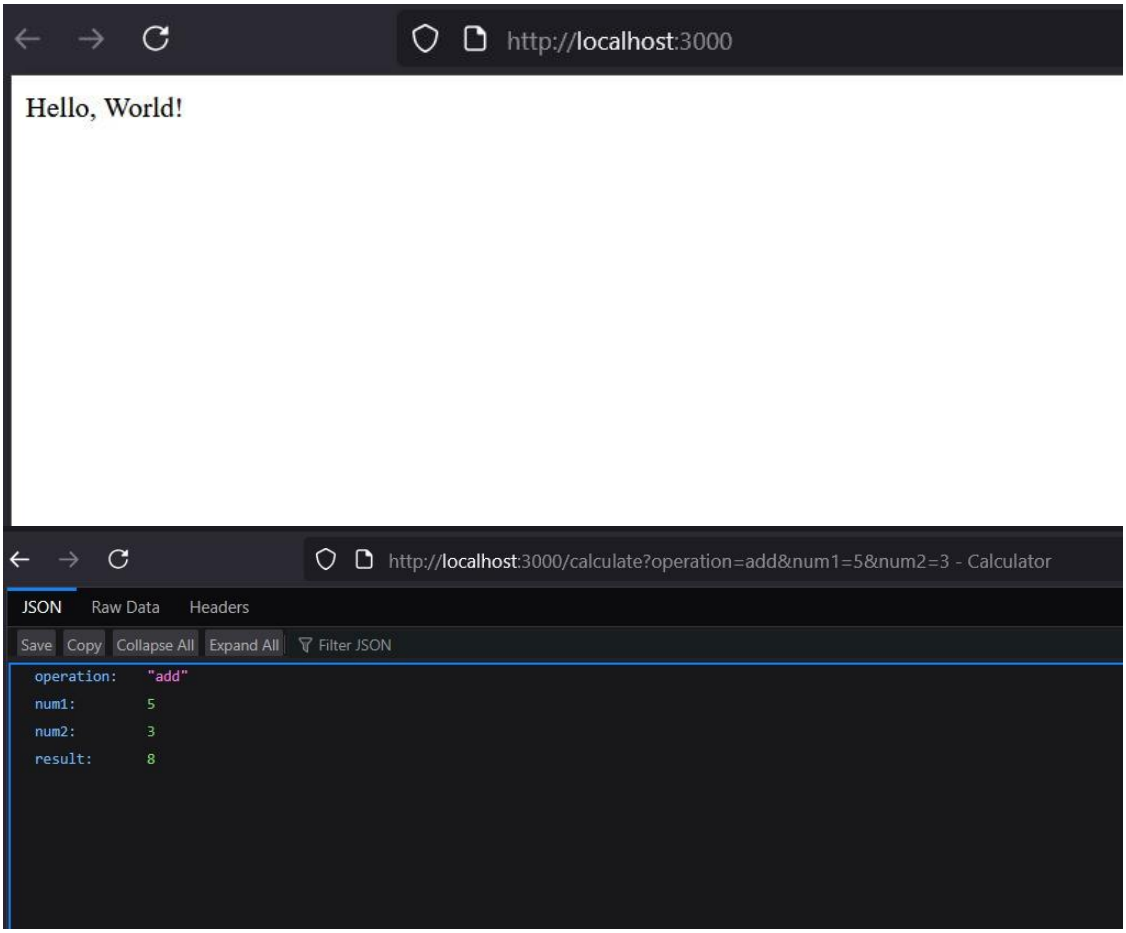
```
      break;
    case 'subtract':
      result = n1 - n2;
      break;
    case 'multiply':
      result = n1 * n2;
      break;
    case 'divide':
      result = n2 !== 0 ? n1 / n2 : 'Error: Division by zero';
      break;
    default:
      return res
        .status(400)
        .json({ error: 'Invalid operation. Use add, subtract, multiply, or di
vide' });
  }

  res.json({ operation, num1: n1, num2: n2, result });
});

// Array iteration endpoint
app.get('/iterate', (req, res) => {
  const array = [10, 20, 30, 40, 50];
  const iterations = [];

  // Using for loop
  iterations.push("Using for loop:");
  for (let i = 0; i < array.length; i++) {
    iterations.push(`Index ${i}: ${array[i]}`);
  }

  // Using forEach
  iterations.push("Using forEach:");
  array.forEach((item, index) => {
    iterations.push(`Index ${index}: ${item}`);
  });

  // Using for...of
  iterations.push("Using for...of:");
  for (const item of array) {
    iterations.push(`Item: ${item}`);
  }

  res.json({ array, iterations });
});

app.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}`);
  console.log('Available endpoints:');
```

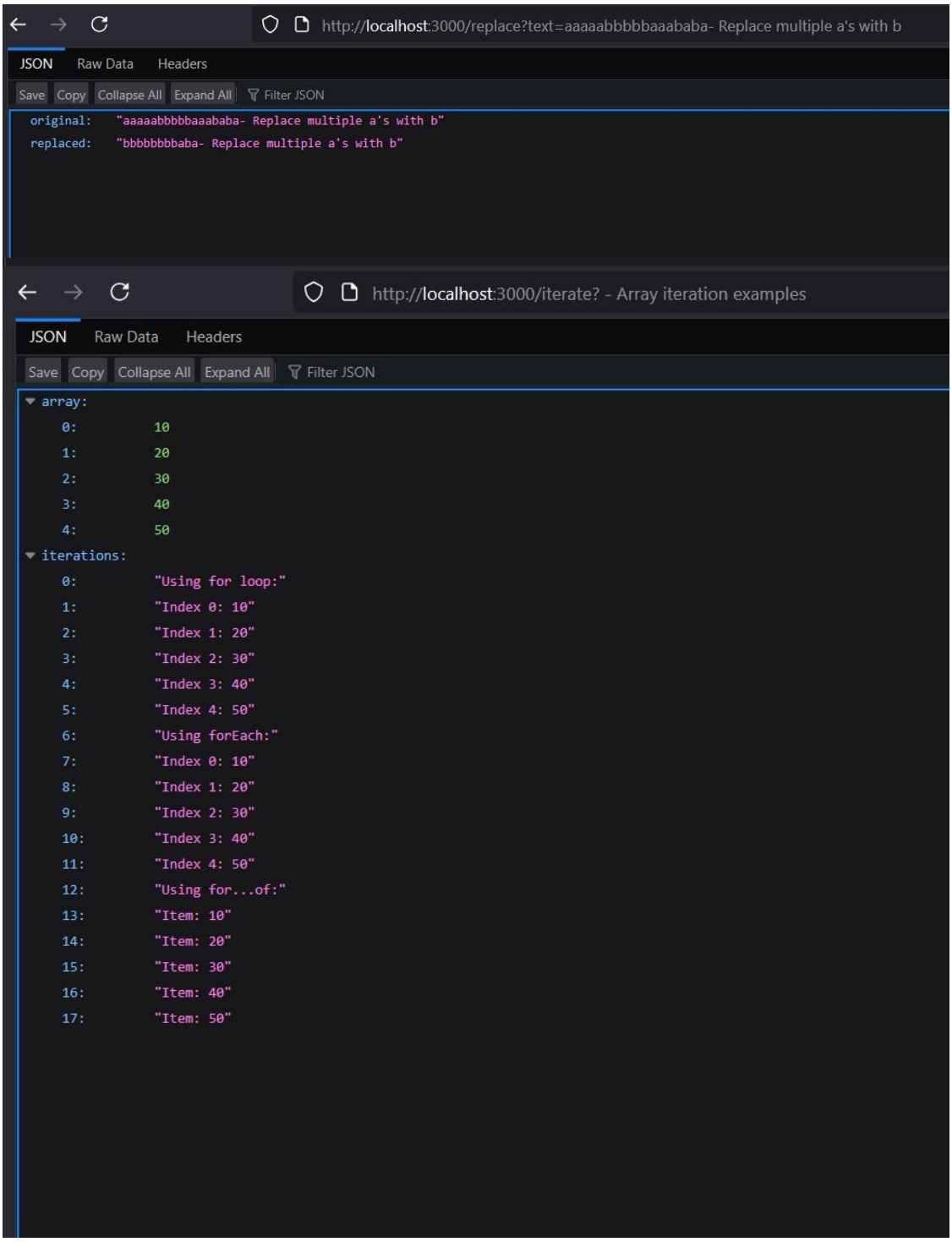
```
  console.log('  GET / - Hello World');
  console.log("  GET /replace?text=your_text - Replace multiple a's with b");
  console.log('  GET /calculate?operation=add&num1=5&num2=3 - Calculator');
  console.log('  GET /iterate - Array iteration examples');
});
```

package.json

```
{
  "name": "express",
  "version": "1.0.0",
  "description": "testing nodejs",
  "main": "server.js",
  "scripts": {
    "test": "node server.js",
    "start": "node server.js"
  },
  "repository": {
    "type": "git",
    "url": "awt01"
  },
  "keywords": [
    "[\"nodejs\"",
    "\"express\"",
    "\"lab\"]"
  ],
  "author": "neha",
  "license": "ISC"
}
```

output


http://localhost:3000
Hello, World!
http://localhost:3000/calculate?operation=add&num1=5&num2=3 - Calculator
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
operation: "add"
num1: 5
num2: 3
result: 8



Challenges Faced

- Installing and configuring required packages using *npm*
- Managing file paths correctly (especially on Windows)

- Understanding the *request-response cycle*

Experiment 7 – Cookies and Sessions

What I Learned

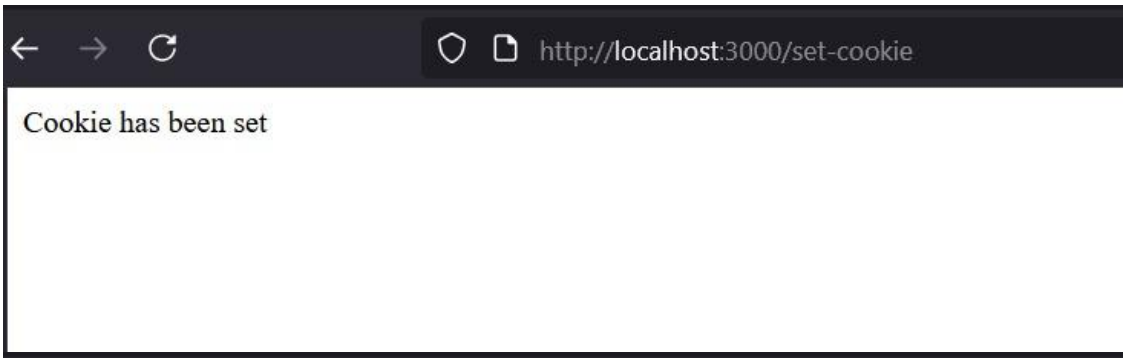
- Setting and reading *cookies* in Node.js
- Managing *sessions* to store user information
- Understanding how *client-server data persistence* works

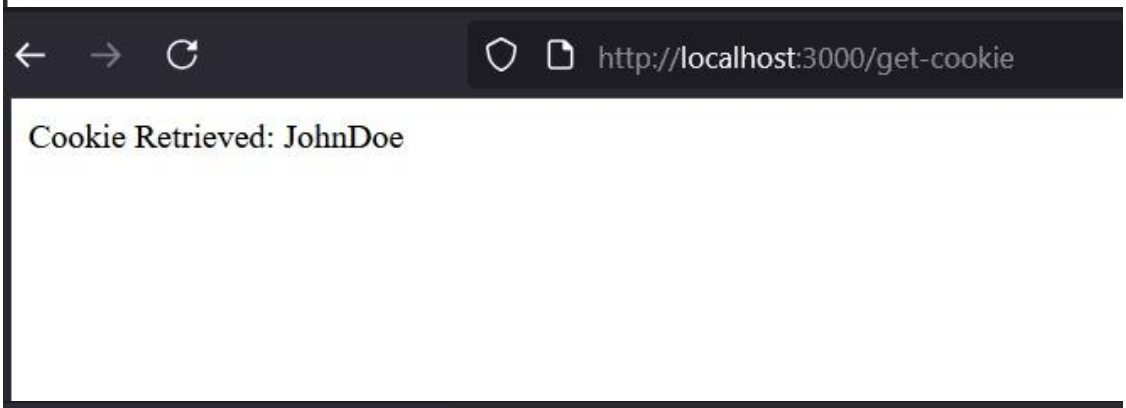
code #### cookie-example.js

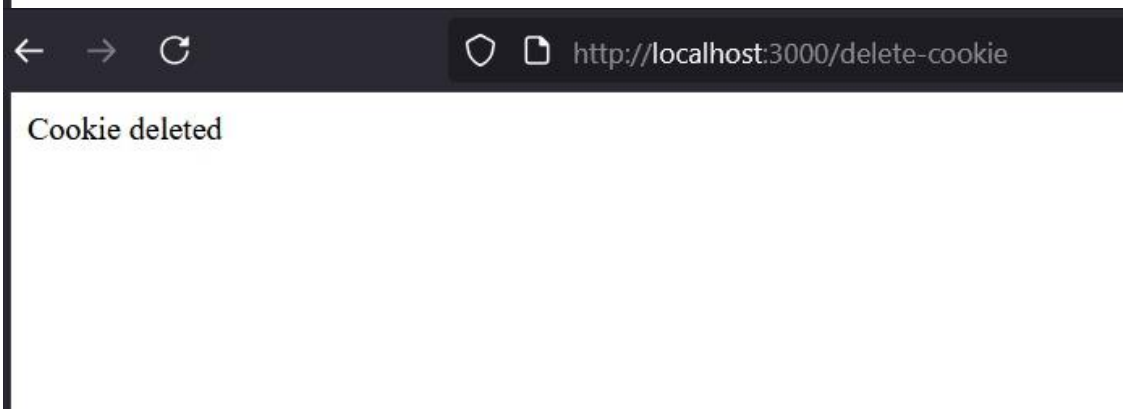
```
 const express = require('express');
const cookieParser = require('cookie-parser');

const app = express();
app.use(cookieParser());

// Set a cookie
app.get('/set-cookie', (req, res) => {
    res.cookie('username', 'JohnDoe', { maxAge: 900000 });
    res.send('Cookie has been set');
});

// Get the cookie
app.get('/get-cookie', (req, res) => {
    const user = req.cookies['username'];
    if (user) {
        res.send(`Cookie Retrieved: ${user}`);
    } else {
        res.send('No cookie found');
    }
});

// Delete the cookie
app.get('/delete-cookie', (req, res) => {
    res.clearCookie('username');
    res.send('Cookie deleted');
});

// Start the server
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Server started on http://localhost:${PORT}`);
});
```

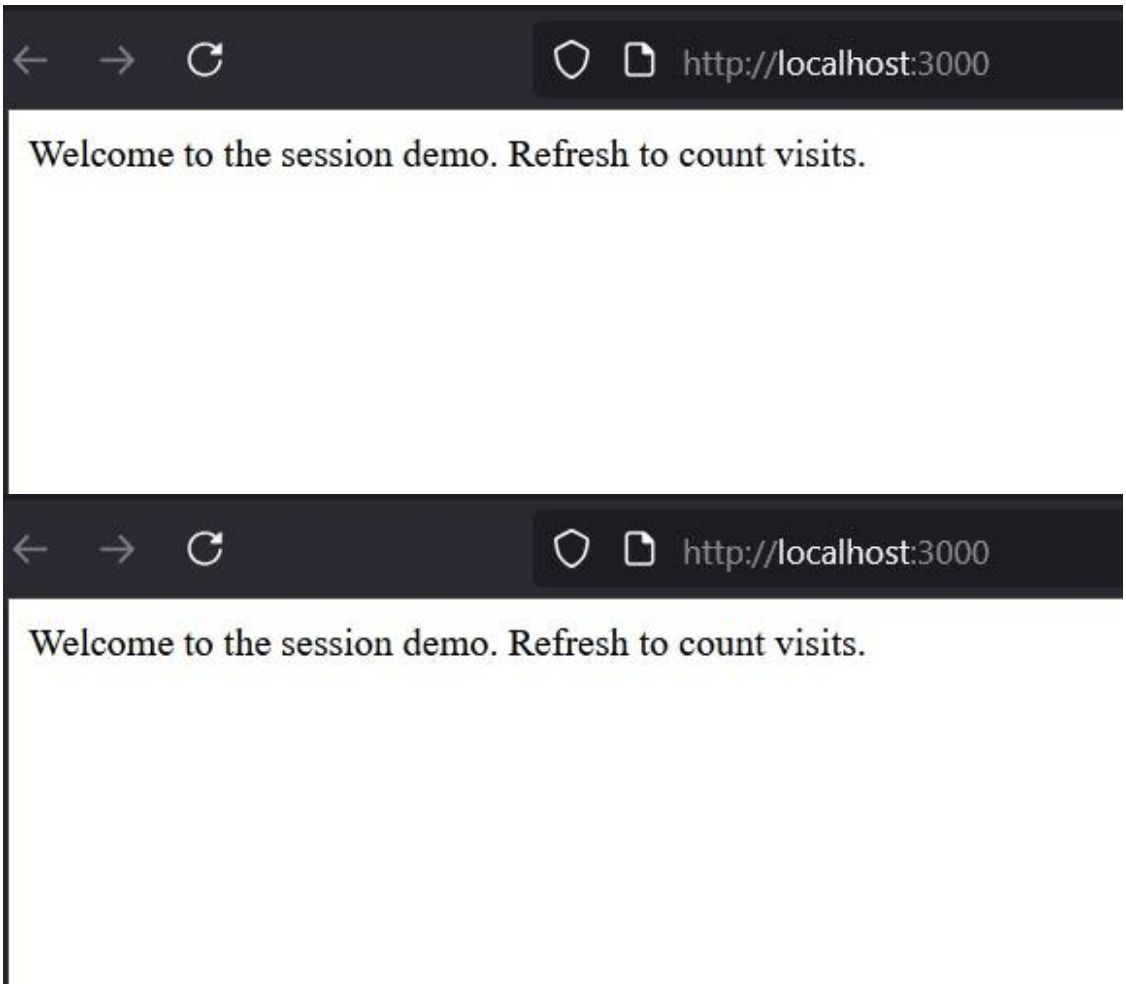
session-example.js

```
const express = require('express');
const session = require('express-session');

const app = express();

app.use(session({
    secret: 'mysecretkey',
    resave: false,
    saveUninitialized: true
}));

app.get('/', (req, res) => {
    if (req.session.views) {
        req.session.views++;
        res.send(`Welcome back! You visited ${req.session.views} times.`);
    } else {
        req.session.views = 1;
        res.send('Welcome to the session demo. Refresh to count visits.');
    }
});

app.get('/destroy', (req, res) => {
    req.session.destroy(err => {
        if (err) {
            return res.send('Error destroying session');
        }
        res.send('Session destroyed');
    });
});

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

app.listen(PORT, () => {
    console.log(`Server started on http://localhost:${PORT}`);
});
```


http://localhost:3000/set-cookie
Cookie has been set


http://localhost:3000/get-cookie
Cookie Retrieved: JohnDoe


http://localhost:3000/delete-cookie
Cookie deleted



Challenges Faced

- Fixing the “Cannot find module” error while running scripts
- Understanding how cookies expire and how session IDs are managed
- Debugging Node.js modules and directory issues

Experiment 8

*Topic: NodeJS + MongoDB Integration, CRUD Operations. Files: item.js, server.js, item.html,s.css ,student.html,student.js.

What I learned: - Creating a NodeJS application to connect to a MongoDB database - Creating an application to store student details in a database - Creating a search application to find students based on criteria - Creating a shopping center application with full CRUD features

Code

shopping-app

item.html

```html
<!DOCTYPE html>
<html>
<head>
  <title>Shopping Center</title>
  <link rel="stylesheet" href="s.css">
</head>
<body>

<h1>Shopping Center Management</h1>

<h2>Add Item</h2>
<form id="addForm">
  <input name="name" placeholder="Item Name" required><br>
  <input name="price" placeholder="Price" type="number" required><br>
  <input name="quantity" placeholder="Quantity" type="number" required><br>
  <button type="submit">Add Item</button>
</form>

<h2>Stock Report</h2>
<table border="1">
  <thead>
    <tr>
      <th>Name</th><th>Price</th><th>Qty</th><th>Actions</th>
    </tr>
  </thead>
  <tbody id="itemTable"></tbody>
</table>

<script src="item.js"></script>
</body>
</html>
```

s.css

```css
/* ---------- GLOBAL ---------- */
body {
  font-family: Arial, Helvetica, sans-serif;
  background: #f4f6f9;
  margin: 0;
  padding: 40px;
}

h1, h2 {
  color: #222;
  margin-bottom: 15px;
```

```
}

h1 {
  font-size: 32px;
  font-weight: 700;
}

h2 {
  font-size: 22px;
  font-weight: 600;
}

/* ---------- FORM AREA ---------- */
form input {
  width: 280px;
  padding: 12px;
  margin: 8px 0;
  border: 1px solid #ccc;
  border-radius: 6px;
  font-size: 16px;
}

button {
  padding: 10px 18px;
  background: #0051a8;
  border: none;
  color: white;
  border-radius: 6px;
  font-size: 15px;
  cursor: pointer;
  margin-right: 8px;
  transition: 0.2s ease;
}

button:hover {
  background: #003d82;
}

/* ---------- TABLE ---------- */
table {
  width: 100%;
  background: #fff;
  margin-top: 20px;
  border-collapse: collapse;
  border-radius: 10px;
  overflow: hidden;
  box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}
```

```css
th {
  background: #0051a8;
  color: #fff;
  padding: 14px;
  font-size: 16px;
  text-align: left;
}

td {
  padding: 14px;
  font-size: 16px;
  border-bottom: 1px solid #eee;
}

/* row hover */
tr:hover {
  background: #f2f7ff;
}

/* buttons inside table */
.action-btn {
  padding: 6px 14px;
  margin-right: 6px;
  border-radius: 5px;
  cursor: pointer;
  border: none;
  font-weight: 500;
}

.delete-btn {
  background: #d9534f;
  color: white;
}

.update-btn {
  background: #0275d8;
  color: white;
}

.sale-btn {
  background: #5cb85c;
  color: white;
}

.delete-btn:hover { background: #c9302c; }
.update-btn:hover { background: #025aa5; }
.sale-btn:hover   { background: #449d44; }

/* ---------- TABLE EMPTY STATE ---------- */
```

```css
.empty {
  text-align: center;
  padding: 20px;
  color: gray;
  font-style: italic;
  font-size: 18px;
}
```

item.js

```js
// Load stock
async function loadItems() {
    let res = await fetch("/items");
    let items = await res.json();

    let table = document.getElementById("itemTable");
    table.innerHTML = "";

    items.forEach(i => {
        table.innerHTML += `
        <tr>
            <td>${i.name}</td>
            <td>${i.price}</td>
            <td>${i.quantity}</td>
            <td>
                <button onclick="deleteItem('${i._id}')">Delete</button>
                <button onclick="updateItem('${i._id}')">Update</button>
                <button onclick="saleItem('${i._id}')">Sale</button>
            </td>
        </tr>`;
    });
}
loadItems();

// Add item
document.getElementById("addForm").addEventListener("submit", async e => {
    e.preventDefault();

    let data = Object.fromEntries(new FormData(e.target).entries());

    await fetch("/items/add", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(data)
    });

    e.target.reset();
    loadItems();
});
```

```
// Delete
async function deleteItem(id) {
    await fetch(`/items/delete/${id}`, { method: "DELETE" });
    loadItems();
}

// Update
async function updateItem(id) {
    let price = prompt("Enter new price:");
    let qty = prompt("Enter new quantity:");

    await fetch(`/items/update/${id}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ price, quantity: qty })
    });

    loadItems();
}

// Sale
async function saleItem(id) {
    let qty = prompt("Enter quantity sold:");

    await fetch(`/items/sale/${id}`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({ quantity: qty })
    });

    loadItems();
}
```

server.js

```
const express = require("express");
const path = require("path");
const mongoose = require("mongoose");

const app = express();
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(__dirname));

// MongoDB Connection
mongoose.connect("mongodb://127.0.0.1:27017/shopDB")
  .then(() => console.log("MongoDB Connected"))
  .catch(err => console.log(err));
```

```
// Schema
const itemSchema = new mongoose.Schema({
  name: String,
  price: Number,
  quantity: Number
});

const Item = mongoose.model("Item", itemSchema);

// Serve UI
app.get("/", (req, res) => {
  res.sendFile(path.join(__dirname, "item.html"));
});

// Add Item
app.post("/items/add", async (req, res) => {
  let item = new Item(req.body);
  await item.save();
  res.json({ success: true });
});

// Get All Items (Stock Report)
app.get("/items", async (req, res) => {
  let items = await Item.find();
  res.json(items);
});

// Delete Item
app.delete("/items/delete/:id", async (req, res) => {
  await Item.findByIdAndDelete(req.params.id);
  res.json({ success: true });
});

// Update Item
app.put("/items/update/:id", async (req, res) => {
  await Item.findByIdAndUpdate(req.params.id, req.body);
  res.json({ success: true });
});

// Sale (reduce quantity)
app.put("/items/sale/:id", async (req, res) => {
  let item = await Item.findById(req.params.id);
  if (!item) return res.json({ success: false });

  let qty = Number(req.body.quantity);
  if (item.quantity < qty)
    return res.json({ success: false, msg: "Not enough stock" });
```

```
  item.quantity -= qty;
  await item.save();

  res.json({ success: true });
});

app.listen(3000, () => console.log("Server running on http://localhost:3000
"));
```

code

student-app

item.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Student Management</title>

  <style>
    body {
      font-family: Arial;
      padding: 30px;
      background: #f5f7fa;
    }
    h2 {
      color: #003366;
      border-left: 5px solid #003366;
      padding-left: 10px;
    }
    input {
      padding: 8px;
      margin: 5px 0;
      width: 250px;
      border: 1px solid #ccc;
      border-radius: 4px;
    }
    button {
      padding: 8px 16px;
      background: #003366;
      color: white;
      border: none;
      border-radius: 4px;
      cursor: pointer;
    }
    button:hover {
      background: #0055a5;
    }
    table {
```

```
      width: 80%;
      margin-top: 15px;
      border-collapse: collapse;
      background: white;
      box-shadow: 0 0 10px #ccc;
    }
    th, td {
      padding: 12px;
      border: 1px solid #ddd;
      text-align: left;
    }
    th {
      background: #003366;
      color: white;
    }
    .highlight {
      background: yellow !important;
      font-weight: bold;
    }
  </style>
</head>

<body>

<h2>Add Student</h2>

<form id="addForm">
  <input type="text" name="name" placeholder="Name" required><br>
  <input type="number" name="roll" placeholder="Roll" required><br>
  <input type="text" name="branch" placeholder="Branch" required><br>
  <input type="number" name="year" placeholder="Year" required><br>
  <button type="submit">Add Student</button>
</form>

<hr><br>

<h2>Search Student</h2>
<input type="text" id="searchBox" placeholder="Enter name or roll" onkeyup="h
ighlightSearch()">
<button onclick="highlightSearch()">Search</button>

<hr><br>

<h2>All Students</h2>

<table>
  <thead>
    <tr>
      <th>Name</th>
```

```
      <th>Roll</th>
      <th>Branch</th>
      <th>Year</th>
    </tr>
  </thead>
  <tbody id="studentTable"></tbody>
</table>

<!-- Single JS file only -->
<script src="student.js"></script>

</body>
</html>
```

student.js

```
// ADD student
document.getElementById("addForm").addEventListener("submit", async function
(e) {
  e.preventDefault();

  let data = Object.fromEntries(new FormData(e.target).entries());

  let res = await fetch("/students/add", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify(data)
  });

  let result = await res.json();
  if (result.success) {
    alert("Student Added!");
    e.target.reset();
    loadStudents();
  }
});

// LOAD all students
async function loadStudents() {
  let res = await fetch("/students");
  let students = await res.json();

  let table = document.getElementById("studentTable");
  table.innerHTML = "";

  students.forEach(s => {
    table.innerHTML += `
      <tr>
        <td>${s.name}</td>
        <td>${s.roll}</td>
```

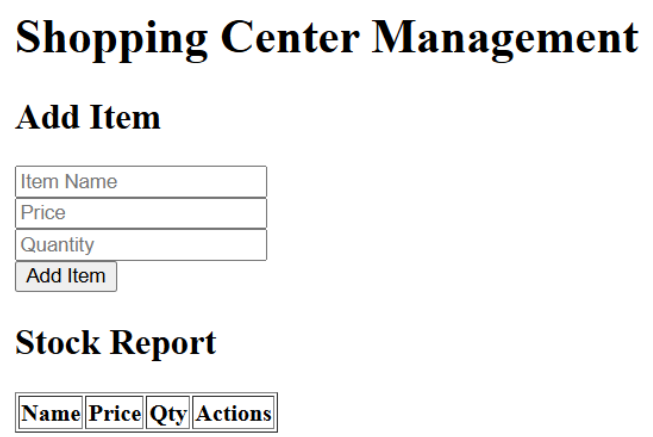
```
        <td>${s.branch}</td>
        <td>${s.year}</td>
      </tr>
    `;
  });
}
loadStudents();

// HIGHLIGHT SEARCH
async function highlightSearch() {
  let keyword = document.getElementById("searchBox").value.toLowerCase();

  let rows = document.querySelectorAll("#studentTable tr");

  rows.forEach(row => row.classList.remove("highlight"));

  if (keyword === "") return;

  rows.forEach(row => {
    let name = row.children[0].textContent.toLowerCase();
    let roll = row.children[1].textContent.toLowerCase();

    if (name.includes(keyword) || roll.includes(keyword)) {
      row.classList.add("highlight");
    }
  });
}
```

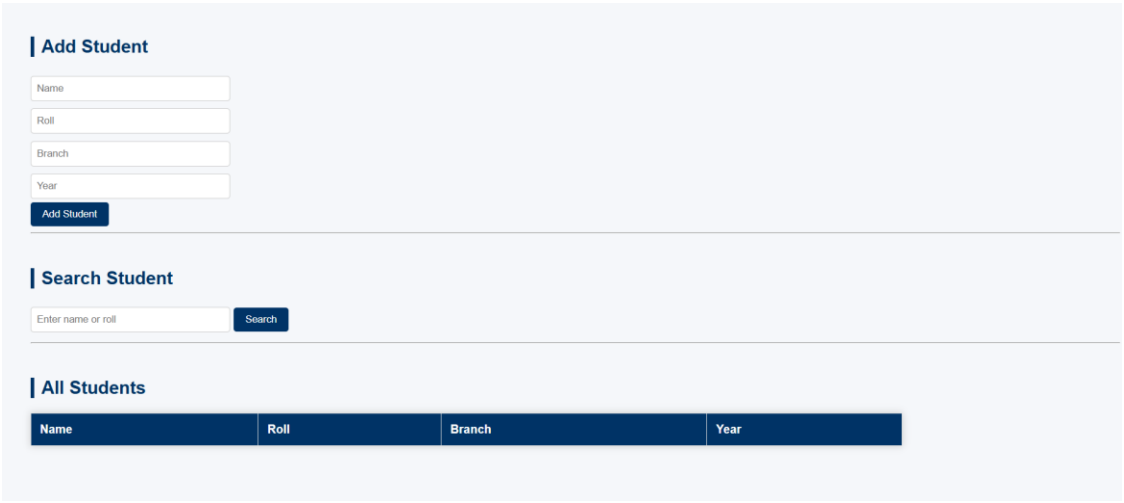
server.js

```
const express = require("express");
const path = require("path");
const mongoose = require("mongoose");

const app = express();

app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(express.static(__dirname));

// MongoDB Connection
mongoose.connect("mongodb://127.0.0.1:27017/studentDB")
  .then(() => console.log("MongoDB Connected"))
  .catch(err => console.log(err));

// Schema
const studentSchema = new mongoose.Schema({
  name: String,
```

```
  roll: Number,
  branch: String,
  year: Number
});

const Student = mongoose.model("Student", studentSchema);

// Serve the HTML page
app.get("/", (req, res) => {
  res.sendFile(path.join(__dirname, "student.html"));
});

// Add student
app.post("/students/add", async (req, res) => {
  try {
    await Student.create(req.body);
    res.json({ success: true });
  } catch (err) {
    res.json({ success: false, error: err });
  }
});

// Get all students
app.get("/students", async (req, res) => {
  const students = await Student.find();
  res.json(students);
});

// Search student (optional)
app.get("/students/search", async (req, res) => {
  const q = req.query.q;

  const result = await Student.find({
    $or: [
      { name: { $regex: q, $options: "i" } },
      { roll: Number(q) }
    ]
  });

  res.json(result);
});

app.listen(3000, () => {
  console.log("Server running on http://localhost:3000");
});
```

output





Challenges faced: - Understanding how async/await works with database operations. - Forgetting to use express.urlencoded() which caused form data to not reach the backend. - Schema mistakes like missing fields or wrong data types. - Difficulty connecting to MongoDB Cloud due to IP access settings.

Experiment 9&10

Topic: SVG Basics, D3.js Visualizations, Interactive Graphics, CSV Data Handling Files: csv.html,data.csv,exp.htm,index.html

What I learned: - Creating a bar chart using SVG and D3.js - Selecting and modifying particular elements using D3 - Creating circles and rectangles as interactive controls - Fetching data from CSV and creating a graph

Code

barchart.html

<!DOCTYPE html>

Bar Chart using D3

Simple Bar Chart

data.csv

name,value A,10 B,30 C,20 D,40

csvchart.html

<!DOCTYPE html>

CSV Data Graph using D3.js

Graph from CSV using SVG & D3.js

index.html

<!DOCTYPE html>

Experiment 9 & 10 - Data Visualization

Experiment 9 & 10: Data Visualization (SVG + D3.js)

Select an experiment:

1. Bar Chart using SVG & D3.js

2. Interactive Circles & Rectangles

3. Select & Modify an SVG Element

4. CSV Data → Graph using SVG & D3.js

modifyelement.html

<!DOCTYPE html>

Modify an SVG Element Using D3

Select and Modify Element

shapes.html

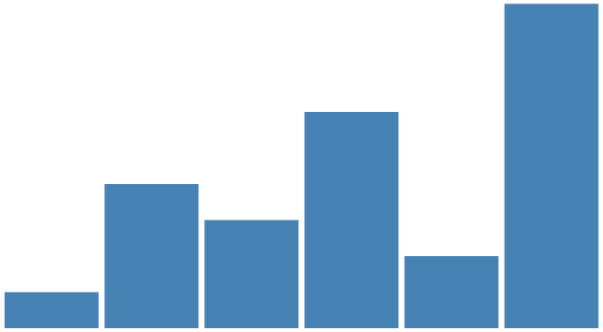
<!DOCTYPE html>

Interactive Shapes

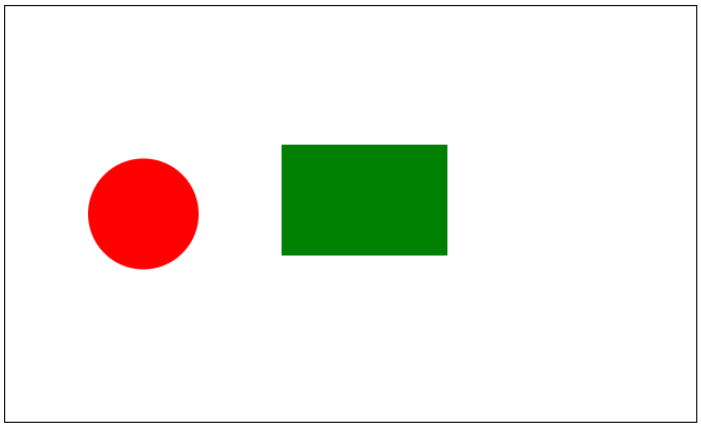
Interactive Circles & Rectangles

output

Simple Bar Chart



Interactive Circles & Rectangles



Select and Modify Element

Graph from CSV using SVG & D3.js

Challenges faced: - Understanding the enter-update-exit pattern of D3.js - Forgetting to include the D3 script link (leading to “d3 is not defined” errors). - Getting confused between pixel values and scale values. -Difficulty in making interactions (click/hover) work correctly.

Final Reflection

This lab gave me practical, hands-on experience in *full-stack web development*.
Each experiment built upon the previous one, helping me connect the dots between *frontend frameworks, **server-side scripting, and** data management*.

Technologies Used

- HTML, CSS, JavaScript
- jQuery, AngularJS
- Node.js, Express.js
- Visual Studio Code
- Git & GitHub for version control

Submitted by: Neha Singh Course: Advanced Web Technology Lab
Tools Used: VS Code, Node.js, GitHub