JASPAL SINGH

590019026

Table of Contents

1. Experiment 1
2. Experiment 2
3. Experiment 3
4. Experiment 4
5. Experiment 5
6. Experiment 5_6
7. Experiment 7
8. Experiment 8
9. Experiment 9&10
10. Conclusion

Experiment 1

Topic: Basic HTML, CSS, and JavaScript Integration
Files: index.html, d1.js, s.css

What I learned: - Creating basic web pages using HTML. - Linking CSS and JS files to an HTML document. - Handling basic DOM manipulation using JavaScript.

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.min.js"></script>
    <link rel="stylesheet" href="s.css">
</head>
```

```
<body>

    <div class="container">
        <h1>1. Right-Click Disabled</h1>
        <p>Try right-clicking anywhere on this page. The context menu
will not 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.png" alt="Scroll to Top" id="scrollToTopBtn">

    <script src="d1.js"></script>
```

```
</body>
</html>
```

s.css

```
body {
            font-family: Arial, sans-serif;
            margin: 20px;
            padding-bottom: 1500px;
        }
        .container {
            margin-bottom: 40px;
        }
        h1 {
            color: orchid;
        }
        p {
            font-size: 1.1em;
            cursor: pointer;
            transition: color 0.3s ease;
        }
        #messageDiv {
            padding: 20px;
            border: 1px solid yellowgreen;
            background-color:white;
            margin-top: 10px;
            border-radius: 8px;
            display: none;
        }
        button {
            padding: 10px 15px;
            margin-right: 10px;
            cursor: pointer;
            border: none;
            border-radius: 5px;
            background-color: pink ;
            color: white;
        }
        button:hover {
            background-color: red;
```

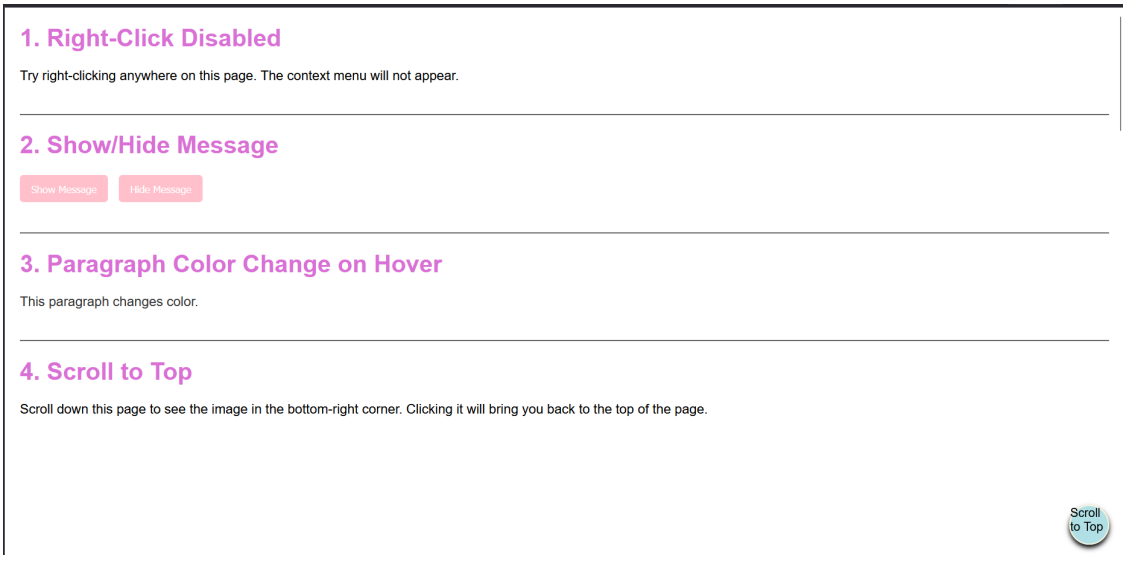
```
        }
        #scrollToTopBtn {
            position: fixed;
            bottom: 20px;
            right: 20px;
            width: 50px;
            height: 50px;
            cursor: pointer;
            border: 2px solid beige;
            border-radius: 50%;
            background-color: powderblue;
            box-shadow: 0 4px 6px black;
        }
```

d1.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: - Understanding how external JS and CSS files are connected. - Debugging syntax errors in JavaScript.

Experiment 2

Topic: Advanced JavaScript – Events and DOM
Files: index.html, d2.js, s.css

What I learned: - Implementing DOM manipulation using JavaScript. - Handling user events like click, hover, etc. - Understanding internal vs external JS files.

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 2</title>
  <link rel="stylesheet" href="s.css">
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <div class="container">
    <h1>Add a class to an element</h1>
    <button id="addbtn">Add border</button>
    <p id="adclass">Click the button to add a border to me</p>
  </div>

  <hr>

  <div class="container">
    <h1>Access element's position</h1>
    <button id="getposition">Get position</button>
    <div id="result"></div>
    <div id="pele">This is the positioned element</div>
  </div>

  <hr>

  <div class="container">
    <h1>Animate multiple CSS properties</h1>
    <button id="animatebtn">Animate box</button>
    <div id="animatedBox"></div>
  </div>

  <script src="d2.js"></script>
```

```
</body>
</html>
```

s.css

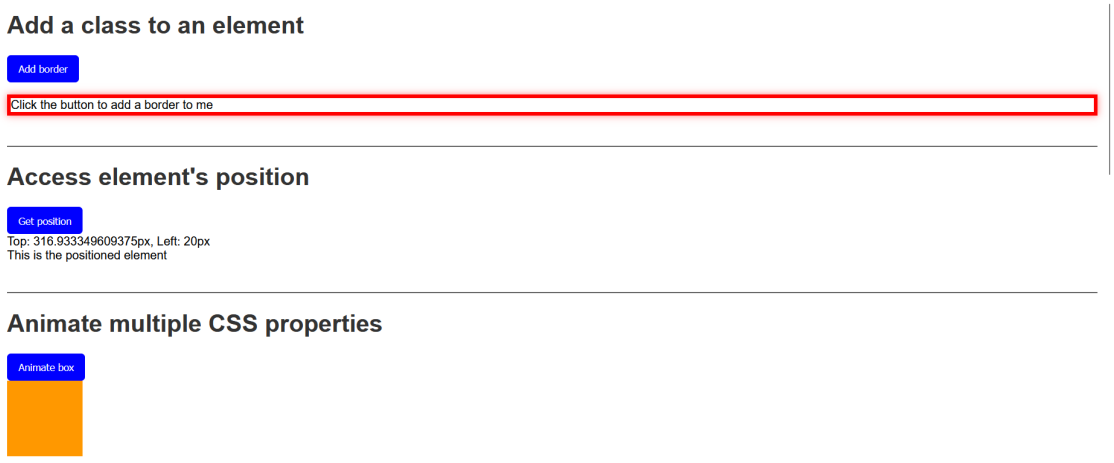
```
body {
  font-family: Arial, sans-serif;
  margin: 20px;
  padding-bottom: 1500px;
}
.container {
  margin-bottom: 40px;
}
h1 {
  color: #333;
}
.highlighted {
  border: 5px solid #ff0000;
  box-shadow: 0 0 10px rgba(255, 0, 0, 0.5);
}
#animatedBox {
  width: 100px;
  height: 100px;
  background-color: #ff9800;
  position: relative;
}
button {
  padding: 10px 15px;
  margin-right: 10px;
  cursor: pointer;
  border: none;
  border-radius: 5px;
  background-color: blue;
  color: white;
}
```

d2.js

```
 $(document).ready(function(){
      $("#addbtn").click(function(){
        $("#adclass").addClass("highlighted");
      });
```

```
      $("#getposition").click(function(){
        var position = $("#pele").position();
        var res = "Top: " + position.top + "px, Left: " +
position.left + "px";
        $("#result").text(res);
      });

      $("#animatebtn").click(function(){
        $("#animatedBox").animate({
          width: '200px',
          height: '200px',
          opacity: 0.5,
          marginLeft: '50px'
        }, 1000, function() {
          $(this).animate({
            width: '100px',
            height: '100px',
            opacity: 1,
            marginLeft: '0px'
          }, 1000);
        });
      });
    });
```

output



Output Screenshot

Challenges faced: - Managing multiple event listeners. - Debugging issues caused by incorrect DOM element references.

Experiment 3

Topic: Introduction to Node.js
Files: index.html, app.js

What I learned: - Setting up a Node.js environment. - Creating a simple web server using Node.js. - Sending responses to client requests.

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="app.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 $index" ng-
class-odd="'odd'" ng-class-even="'even'">
            <td>{{$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>
```

app.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: Age Reverse

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

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: - Installing Node.js packages using npm. - Understanding asynchronous execution in Node.js.

Experiment 4

Topic: HTML Forms and Data Handling
Files: bill.html, form.html

What I learned: - Creating HTML forms and input fields. - Using form attributes like action, method, and name. - Validating form inputs.

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>
```

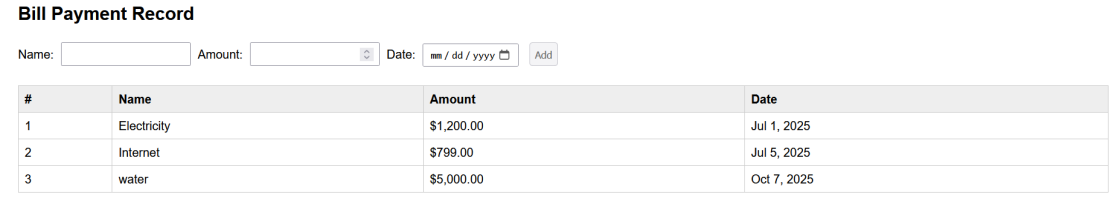
form.html

```
<!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-
minlength="3">
    </label>
    <div class="error"
         ng-show="(regForm.name.$touched || submitted) &&
regForm.name.$invalid">
      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.$touched || submitted) &&
regForm.email.$invalid">
      Enter a valid email.
    </div>
    <br><br>

    <label>Password:
      <input type="password" name="password" ng-model="user.password"
required ng-minlength="6">
    </label>
    <div class="error"
         ng-show="(regForm.password.$touched || submitted) &&
regForm.password.$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>
```

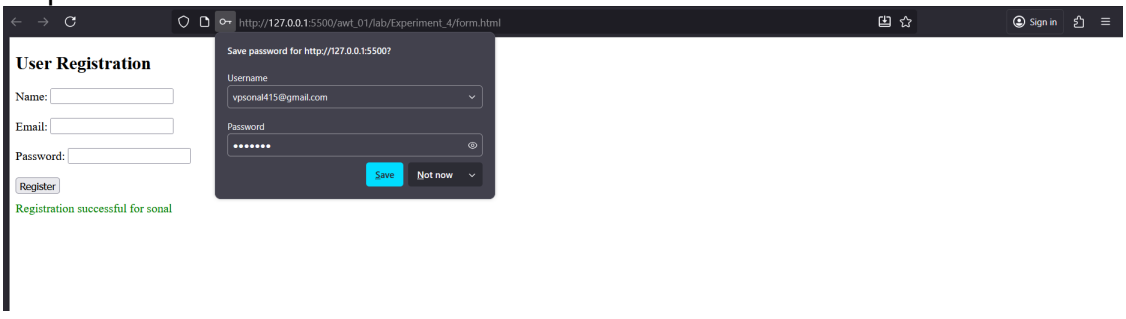
```
  <script>
    angular.module('formApp', [])
      .controller('FormController', function($scope) {
        $scope.user = {};
        $scope.submitted = false;
        $scope.success = '';

        $scope.register = function(form) {
          if (form.$valid) {
            // In real app, send data to server
            $scope.success = 'Registration successful for ' +
$scope.user.name;
            $scope.user = {};
            form.$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



###

output-form



Challenges faced: - Handling form validation using JavaScript. - Designing a clean and structured form layout.

Experiment 5_6

Topic: Express.js and Modular Node.js Applications
Files: server.js, package.json

What I learned: - Installing and using Express.js framework. - Creating routes and handling requests in Express. - Understanding package.json and dependencies.

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 divide' });
  }

  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');
});
```

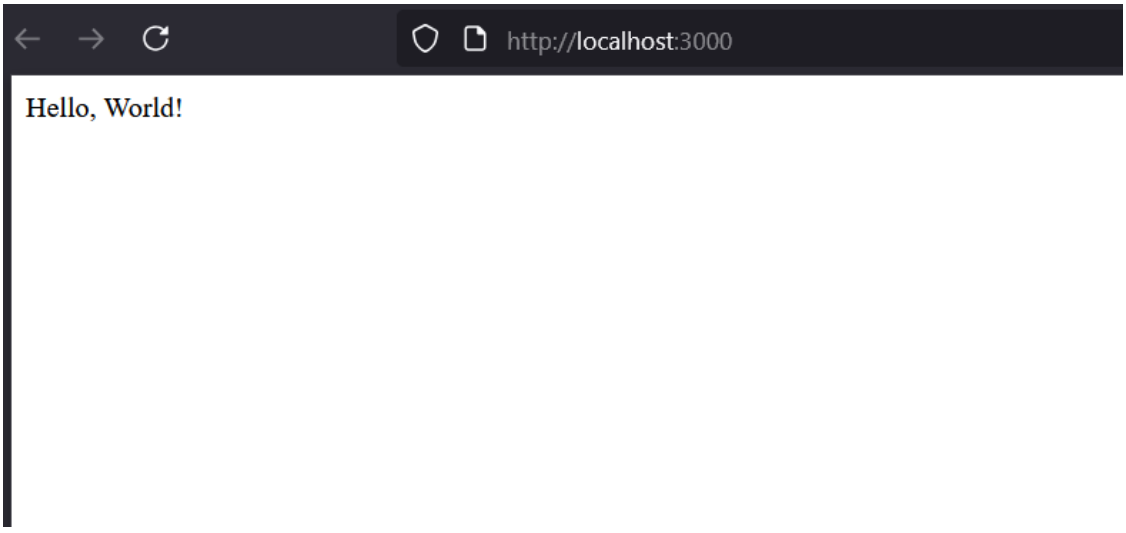
package.json

```
{
  "name": "nodeja-lab",
  "version": "1.0.0",
  "description": "\"NodeJS basic exercies lab\",",
  "main": "index.js",
  "dependencies": {
    "accepts": "^2.0.0",
    "body-parser": "^2.2.0",
    "bytes": "^3.1.2",
    "call-bind-apply-helpers": "^1.0.2",
    "call-bound": "^1.0.4",
    "content-disposition": "^1.0.0",
    "content-type": "^1.0.5",
    "cookie": "^0.7.2",
    "cookie-signature": "^1.2.2",
    "debug": "^4.4.3",
    "depd": "^2.0.0",
    "dunder-proto": "^1.0.1",
    "ee-first": "^1.1.1",
    "encodeurl": "^2.0.0",
    "es-define-property": "^1.0.1",
    "es-errors": "^1.3.0",
    "es-object-atoms": "^1.1.1",
    "escape-html": "^1.0.3",
    "etag": "^1.8.1",
    "express": "^5.1.0",
    "finalhandler": "^2.1.0",
    "forwarded": "^0.2.0",
    "fresh": "^2.0.0",
    "function-bind": "^1.1.2",
    "get-intrinsic": "^1.3.0",
    "get-proto": "^1.0.1",
    "gopd": "^1.2.0",
```

```
    "has-symbols": "^1.1.0",
    "hasown": "^2.0.2",
    "http-errors": "^2.0.0",
    "iconv-lite": "^0.6.3",
    "inherits": "^2.0.4",
    "ipaddr.js": "^1.9.1",
    "is-promise": "^4.0.0",
    "math-intrinsics": "^1.1.0",
    "media-typer": "^1.1.0",
    "merge-descriptors": "^2.0.0",
    "mime-db": "^1.54.0",
    "mime-types": "^3.0.1",
    "ms": "^2.1.3",
    "negotiator": "^1.0.0",
    "object-inspect": "^1.13.4",
    "on-finished": "^2.4.1",
    "once": "^1.4.0",
    "parseurl": "^1.3.3",
    "path-to-regexp": "^8.3.0",
    "proxy-addr": "^2.0.7",
    "qs": "^6.14.0",
    "range-parser": "^1.2.1",
    "raw-body": "^3.0.1",
    "router": "^2.2.0",
    "safe-buffer": "^5.2.1",
    "safer-buffer": "^2.1.2",
    "send": "^1.2.0",
    "serve-static": "^2.2.0",
    "setprototypeof": "^1.2.0",
    "side-channel": "^1.1.0",
    "side-channel-list": "^1.0.0",
    "side-channel-map": "^1.0.1",
    "side-channel-weakmap": "^1.0.2",
    "statuses": "^2.0.2",
    "toidentifier": "^1.0.1",
    "type-is": "^2.0.1",
    "unpipe": "^1.0.0",
    "vary": "^1.1.2",
    "wrappy": "^1.0.2"
  },
```

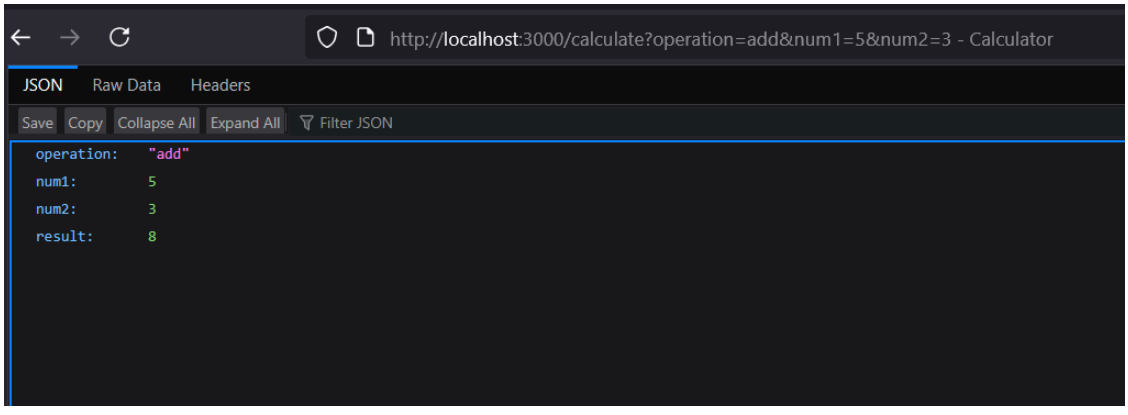
```
  "devDependencies": {},
  "scripts": {
    "test": "node server.js",
    "start": "node lab/Experiment_5_6/server.js"
  },
  "repository": {
    "type": "git",
    "url": "awt_01"
  },
  "keywords": [
    "[\"nodejs\"",
    "\"express\"",
    "\"lab\"]"
  ],
  "author": "sonal",
  "license": "ISC"
}
```

output

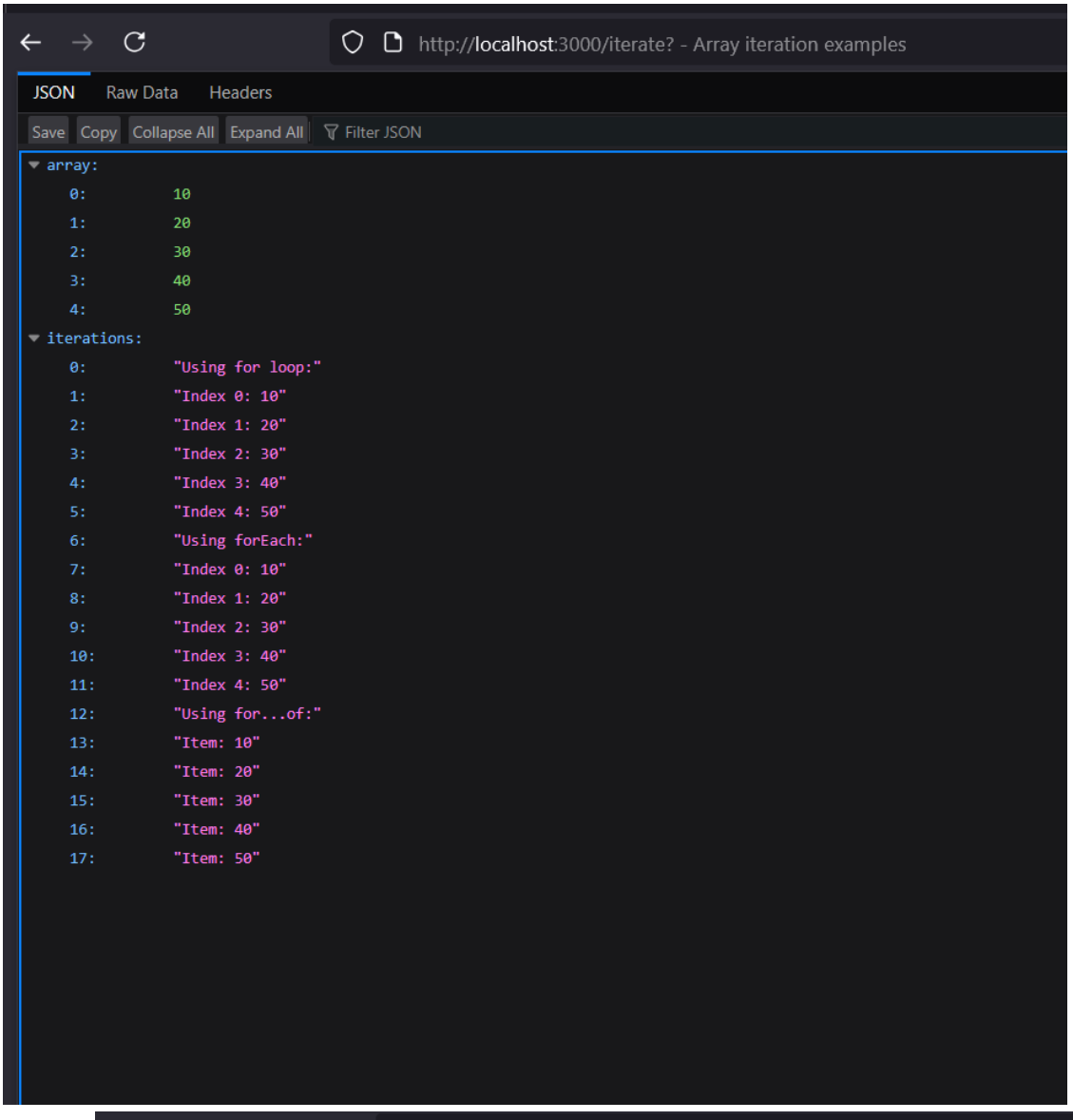


###

output

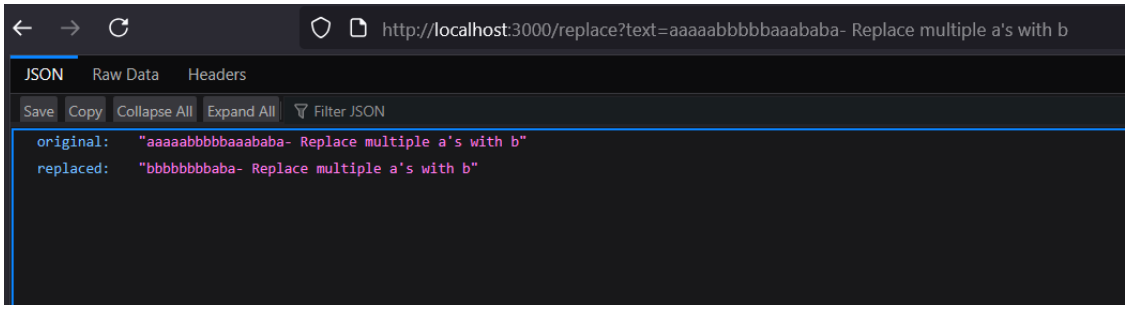


output


http://localhost:3000/iterate? - Array iteration examples
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
array:
0: 10
1: 20
2: 30
3: 40
4: 50
iterations:
0: "Using for loop:"
1: "Index 0: 10"
2: "Index 1: 20"
3: "Index 2: 30"
4: "Index 3: 40"
5: "Index 4: 50"
6: "Using forEach:"
7: "Index 0: 10"
8: "Index 1: 20"
9: "Index 2: 30"
10: "Index 3: 40"
11: "Index 4: 50"
12: "Using for...of:"
13: "Item: 10"
14: "Item: 20"
15: "Item: 30"
16: "Item: 40"
17: "Item: 50"

###

output


http://localhost:3000/replace?text=aaaaabbbbbaaababa- Replace multiple a's with b
JSON
Raw Data
Headers
Save
Copy
Collapse All
Expand All
Filter JSON
original: "aaaaabbbbbaaababa- Replace multiple a's with b"
replaced: "bbbbbbbbaba- Replace multiple a's with b"

Challenges faced: - Configuring Express properly. - Managing multiple JavaScript files in one project.

Experiment 7

Topic: Node.js Sessions and Cookies
Files: source/, server.js, package.json

What I learned: - Using cookies and sessions in Node.js. - Maintaining user sessions across multiple pages. - Understanding middleware and session storage.

Code

cookie-example.js

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

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

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

app.get('/get-cookie', (req, res) => {
    const user = req.cookies['username'];
    res.send(`Cookie Retrieved: ${user}`);
});

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

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

app.listen(PORT, () => {
```

```
    console.log(`Server started on http://localhost:${PORT}`);
});
```

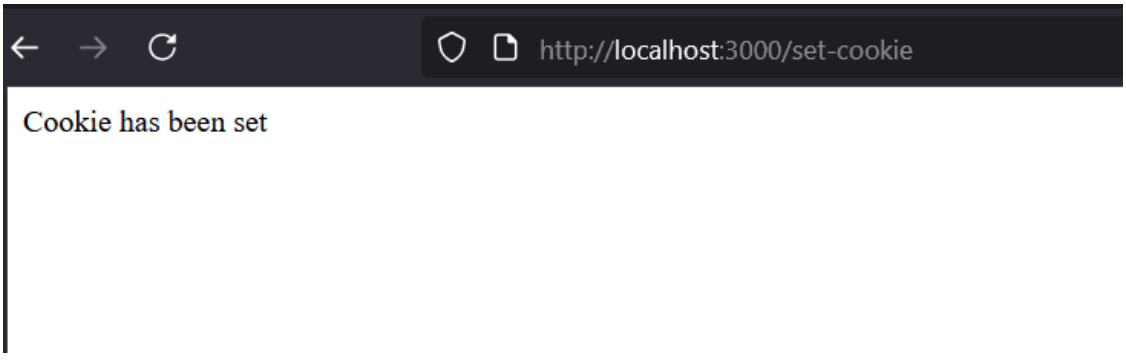
session-example.js

```
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}`);
});
```

package.json

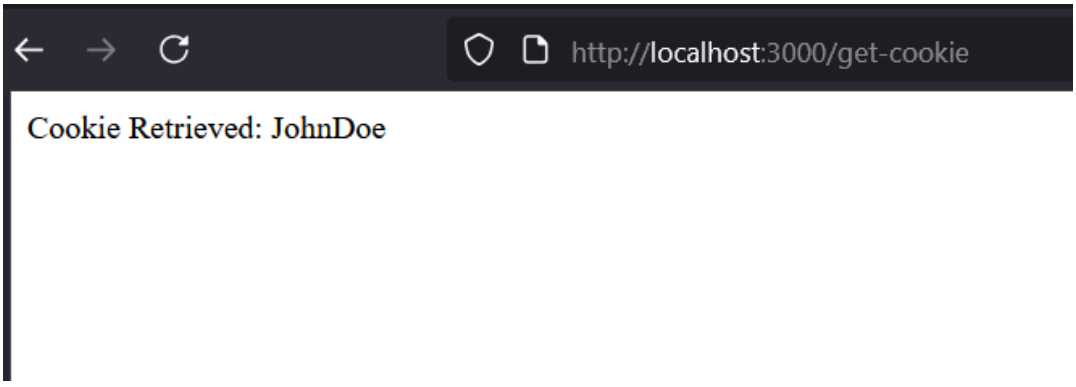
```
{
  "name": "experiment_7",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cookie-parser": "^1.4.7",
    "express": "^5.1.0",
    "express-session": "^1.18.2"
  }
}
```

output-cookie.js

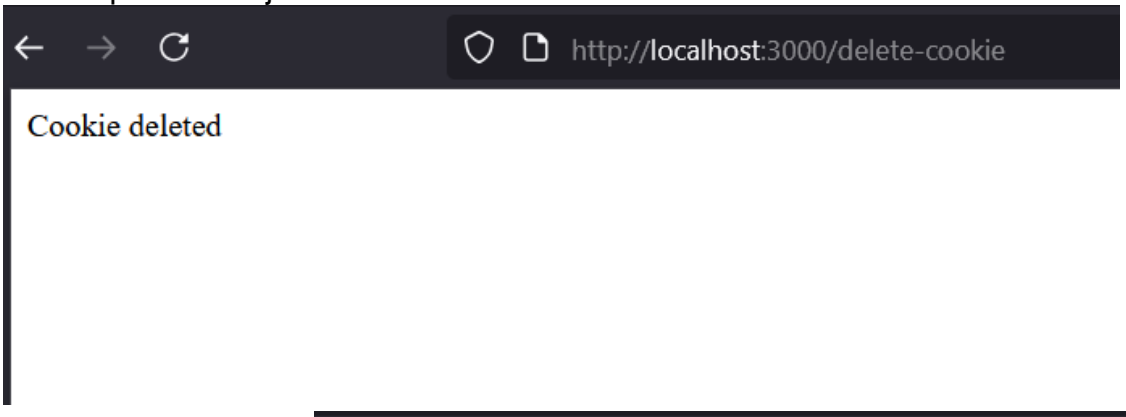


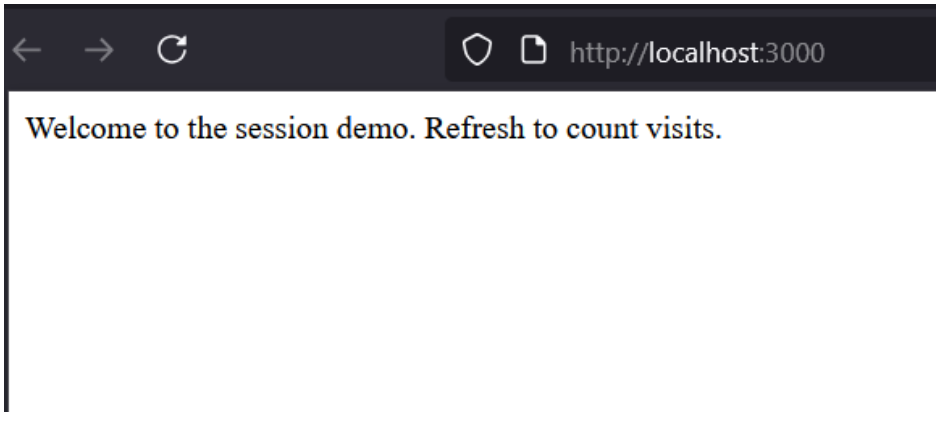
###

output-cookie.js



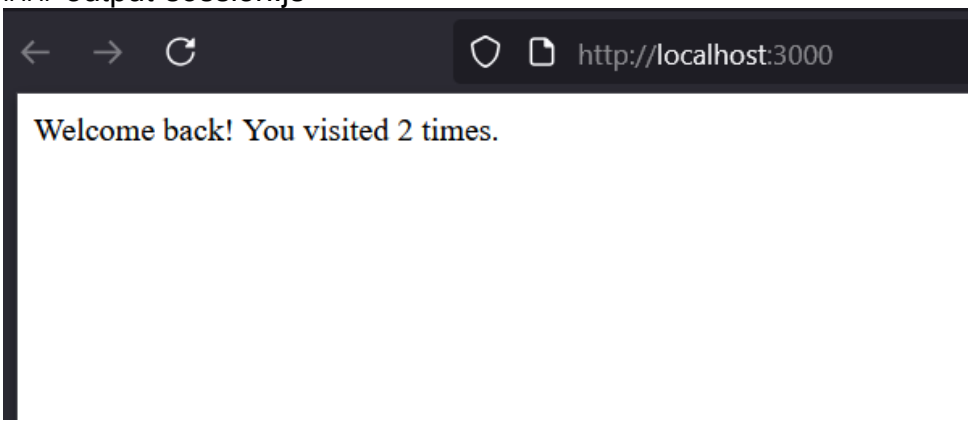
output-cookie.js





output-session.js

output-session.js



Challenges faced: - Configuring session middleware correctly. - Debugging cookie handling 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

```
<!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

```
/* ---------- 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);
}

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

item.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="highlightSearch()">
<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");
    }
  });
}
```

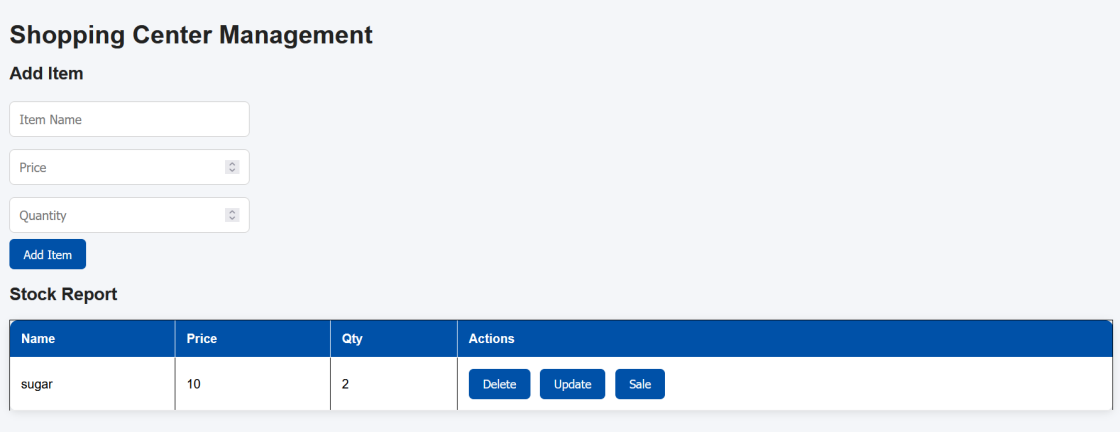
server.js

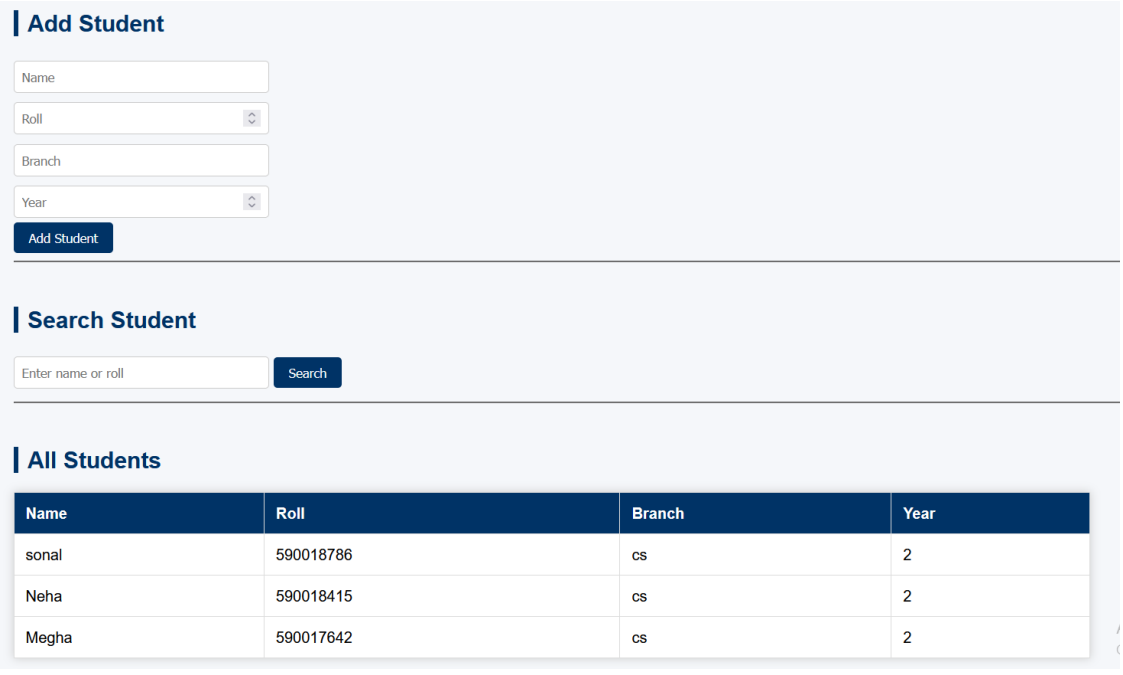
```
const express = require("express");
const path = require("path");
const mongoose = require("mongoose");

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

csv.html

```
<!DOCTYPE html>
<html>
<head>
<title>Experiment 10: CSV Graph</title>
<script src="https://d3js.org/d3.v7.min.js"></script>
</head>

<body>

<h1>Experiment 10: CSV → Graph using D3.js</h1>

<svg width="500" height="300"></svg>

<script>
const svg = d3.select("svg");
const width = 500;
const height = 300;
const barWidth = 50;

d3.csv("data.csv").then(data => {

  data.forEach(d => d.value = +d.value);

  svg.selectAll("rect")
    .data(data)
```

```
    .enter()
    .append("rect")
    .attr("x", (d, i) => i * (barWidth + 10))
    .attr("y", d => height - d.value * 5)
    .attr("width", barWidth)
    .attr("height", d => d.value * 5)
    .attr("fill", "teal");

  svg.selectAll("text")
    .data(data)
    .enter()
    .append("text")
    .text(d => d.name)
    .attr("x", (d, i) => i * (barWidth + 10) + 15)
    .attr("y", height - 5)
    .attr("fill", "black");
});
</script>

</body>
</html>
```

data.csv

```
name,value
A,10
B,20
C,15
D,25
```

exp.html

```
<!DOCTYPE html>
<html>
<head>
<title>Experiment 9: Combined Visualizations</title>
<script src="https://d3js.org/d3.v7.min.js"></script>

<style>
section {
```

```
  margin-bottom: 50px;
  padding: 20px;
  border: 2px solid #ddd;
  border-radius: 10px;
}
h2 { color: #003366; }
button { padding: 8px 15px; margin-right: 10px; }
</style>
</head>

<body>

<h1>Experiment 9: SVG + D3 (Combined)</h1>

<!-- ========================================================= -->
<!-- =============== 1. BAR CHART ============================ -->
<!-- ========================================================= -->
<section>
<h2>1. Bar Chart using SVG + D3.js</h2>
<svg id="barChart" width="500" height="250"></svg>

<script>
const data = [10, 30, 20, 40, 25];

const svg1 = d3.select("#barChart");
const height1 = 250;
const barWidth = 50;

svg1.selectAll("rect")
  .data(data)
  .enter()
  .append("rect")
  .attr("x", (d, i) => i * (barWidth + 10))
  .attr("y", d => height1 - d * 5)
  .attr("width", barWidth)
  .attr("height", d => d * 5)
  .attr("fill", "steelblue");
</script>
</section>
```

```
<!-- ========================================================= -->
<!-- ================ 2. INTERACTIVE SHAPES ================== -->
<!-- ========================================================= -->
<section>
<h2>2. Interactive Shapes (Circles & Rectangles)</h2>

<svg id="shapes" width="500" height="250"></svg>

<script>
const svg2 = d3.select("#shapes");

// Circle
svg2.append("circle")
  .attr("cx", 100)
  .attr("cy", 120)
  .attr("r", 40)
  .attr("fill", "orange")
  .on("mouseover", () => d3.select("circle").attr("fill", "red"))
  .on("mouseout", () => d3.select("circle").attr("fill", "orange"));

// Rectangle
svg2.append("rect")
  .attr("x", 200)
  .attr("y", 100)
  .attr("width", 80)
  .attr("height", 60)
  .attr("fill", "green")
  .on("click", function(){ d3.select(this).attr("fill", "purple"); });
</script>
</section>


<!-- ========================================================= -->
<!-- ============ 3. MODIFY ELEMENT USING D3 ============= -->
<!-- ========================================================= -->
<section>
<h2>3. Modify Selected Element</h2>

<svg id="mod" width="400" height="200">
```

```
  <circle id="myCircle" cx="150" cy="100" r="50" fill="blue"></circle>
</svg>

<button onclick="changeColor()">Change Color</button>
<button onclick="increaseSize()">Increase Size</button>

<script>
function changeColor() {
  d3.select("#myCircle").attr("fill", "red");
}

function increaseSize() {
  d3.select("#myCircle").attr("r", 80);
}
</script>

</section>

</body>
</html>
```

index.html

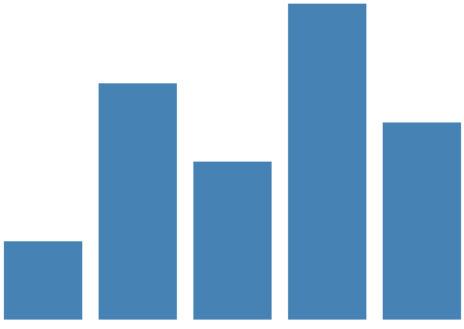
```
<!DOCTYPE html>
<html>
<head>
<title>Data Visualization Experiments</title>

<style>
body {
  font-family: Arial;
  padding: 40px;
  background: #f5f7fa;
}
.box {
  border: 2px solid #003366;
  padding: 20px;
  border-radius: 8px;
  width: 350px;
  margin-bottom: 20px;
```

```
  background: white;
}
a {
  display: block;
  text-decoration: none;
  color: white;
  background: #003366;
  padding: 10px;
  text-align: center;
  border-radius: 6px;
}
a:hover { background: #0055a5; }
</style>
</head>

<body>

<h1>Data Visualization - Experiment 9 & 10</h1>

<div class="box">
  <h3>Experiment 9 (Combined)</h3>
  <p>Bar Chart + Interactive Shapes + Modify SVG Element</p>
  <a href="exp.html">Open Experiment 9</a>
</div>

<div class="box">
  <h3>Experiment 10</h3>
  <p>CSV → Graph using D3</p>
  <a href="csv.html">Open Experiment 10</a>
</div>

</body>
</html>
```

output

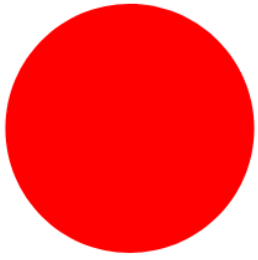
Experiment 9: SVG + D3 (Combined)

1. Bar Chart using SVG + D3.js



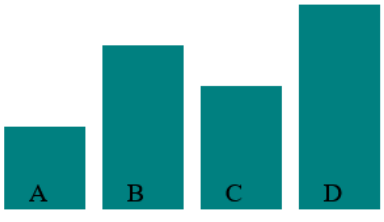
2. Interactive Shapes (Circles & Rectangles)

3. Modify Selected Element



Change Color Increase Size

Experiment 10: CSV → Graph using 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.

Conclusion

This lab helped me gain hands-on experience in *front-end and back-end web development*.
I learned how to: - Design interactive front-end applications (HTML, CSS, JS, jQuery, Angular). - Build and deploy backend applications using *Node.js and Express*. - Handle user sessions, cookies, and server routing.

Overall Challenge:
Initially, understanding how client and server communicate was difficult, but through these experiments, I developed a clear understanding of *full-stack web development*.

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