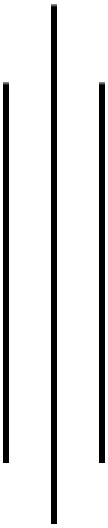
TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS



WEB -PROGRAMMING AND APPLICATION

Submitted by :

Name : Avinash kumar yadav
Roll No. : 080bct018
Group: A

Submitted to :

Department of Electronics
and Computer Engineering,
Pulchowk Campus

Lab Report

Assignment 1: Smart Calculator with History

1. Problem Understanding

The objective of this assignment is to design a **Smart Calculator** using **Vanilla JavaScript** that can perform basic arithmetic operations such as addition, subtraction, multiplication, and division.

The calculator should:

- Store the **last 5 calculations**
- Display calculation history **dynamically**
- Allow the user to **clear the history**
- Handle **invalid inputs** such as division by zero
- Use an **array** to store the calculation history

No external libraries or frameworks are allowed.

2. List of JavaScript Concepts Used

- Variables (let, const)
- Functions
- Arrays
- Conditional statements (if, else)
- DOM Manipulation
    - document.getElementById()
    - innerHTML
- Event handling (onclick)
- String concatenation

- Basic arithmetic operators

3. Flow Diagram / Logic Steps

1. User enters two numbers
2. User selects an operation (+, −, ×, ÷)
3. JavaScript function is called
4. Input validation is performed
    - Check for empty values
    - Check for divide by zero
5. Result is calculated
6. Calculation is stored in an array
7. If history exceeds 5 items, remove the oldest entry
8. Display result and updated history
9. User can clear history using a button

4. Source Code

HTML + JavaScript Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Smart Calculator</title>
  <style>
    body {
```

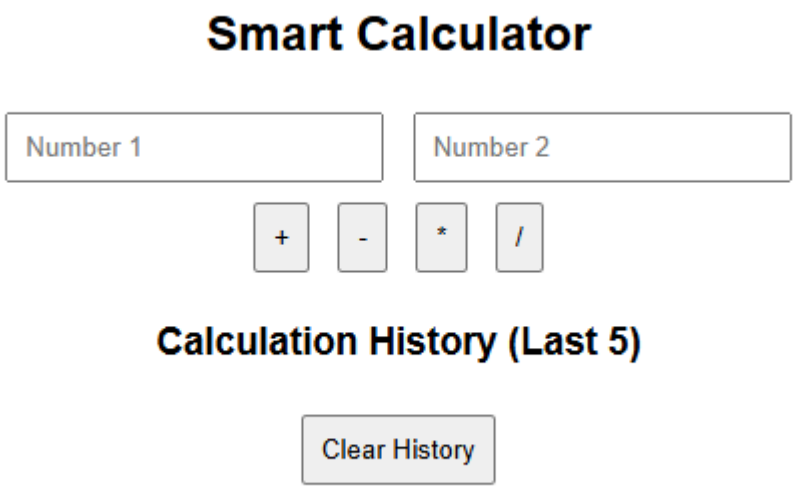
```
        font-family: Arial, sans-serif;
        text-align: center;
        margin-top: 50px;
      }
      input, button {
        padding: 8px;
        margin: 5px;
      }
      #history {
        margin-top: 20px;
      }
    </style>
</head>
<body>

<h2>Smart Calculator</h2>

<input type="number" id="num1" placeholder="Number 1">
<input type="number" id="num2" placeholder="Number 2"><br>

<button onclick="calculate('+')">+</button>
<button onclick="calculate('-')">-</button>
```

```
<button onclick="calculate('*')">*</button>

<button onclick="calculate('/')">/</button>


<h3 id="result"></h3>


<h3>Calculation History (Last 5)</h3>

<div id="history"></div>


<button onclick="clearHistory()">Clear History</button>


<script>

   let history = [];


   function calculate(operator) {

      let n1 = document.getElementById("num1").value;

      let n2 = document.getElementById("num2").value;


      if (n1 === "" || n2 === "") {

         alert("Please enter both numbers");

         return;

      }
```

```
n1 = Number(n1);
n2 = Number(n2);
let result;

if (operator === "/" && n2 === 0) {
   alert("Division by zero is not allowed");
   return;
}

if (operator === "+") result = n1 + n2;
else if (operator === "-") result = n1 - n2;
else if (operator === "*") result = n1 * n2;
else if (operator === "/") result = n1 / n2;

let calculation = `${n1} ${operator} ${n2} = ${result}`;

history.push(calculation);

if (history.length > 5) {
   history.shift();
}
```

```
        document.getElementById("result").innerText = "Result: " + result;
        displayHistory();
    }

    function displayHistory() {
        let historyDiv = document.getElementById("history");
        historyDiv.innerHTML = "";

        for (let i = 0; i < history.length; i++) {
            historyDiv.innerHTML += history[i] + "<br>";
        }
    }

    function clearHistory() {
        history = [];
        document.getElementById("history").innerHTML = "";
        document.getElementById("result").innerText = "";
    }
</script>

</body>
</html>
```

5. Output Screenshots



Conclusion

This assignment successfully demonstrates the use of **Vanilla JavaScript** to build a smart calculator with dynamic history management, proper input validation, and clean DOM manipulation without using any external libraries.

Assignment 2: Student Marks & Grade Analyzer

1. Problem Understanding

The aim of this assignment is to create a **Student Marks & Grade Analyzer** using **Vanilla JavaScript**.
The system should store student details using **objects**, accept marks for **multiple subjects**, calculate **total marks, percentage, and grade**, and display the results in a **table format**. Additionally, the table rows should be **highlighted dynamically** using JavaScript and CSS to indicate whether a student has **passed or failed**.

2. List of JavaScript Concepts Used

- Objects
- Arrays
- Functions

- Conditional statements (if, else)
- Loops (for)
- DOM Manipulation
    - document.getElementById()
    - innerHTML
- Event handling (onclick)
- Arithmetic operations
- Dynamic CSS styling using JavaScript

3. Flow Diagram / Logic Steps

1. User enters student name and marks of subjects
2. Marks are stored in an array
3. Student details are stored as an object
4. Total marks are calculated
5. Percentage is calculated
6. Grade is assigned based on percentage
7. Pass/Fail status is determined
8. Result is added to a table
9. Table row color changes based on pass/fail

4. Source Code

HTML + JavaScript Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Student Marks & Grade Analyzer</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 40px;
        }
        input, button {
            padding: 6px;
            margin: 5px;
        }
        table {
            width: 80%;
            margin: auto;
            border-collapse: collapse;
            margin-top: 20px;
        }
        table, th, td {
```

```
        border: 1px solid black;
    }
    th, td {
        padding: 8px;
    }
    .pass {
        background-color: #c8f7c5;
    }
    .fail {
        background-color: #f7c5c5;
    }
  </style>
</head>
<body>

<h2>Student Marks & Grade Analyzer</h2>

<input type="text" id="name" placeholder="Student Name"><br>
<input type="number" id="sub1" placeholder="Subject 1 Marks">
<input type="number" id="sub2" placeholder="Subject 2 Marks">
<input type="number" id="sub3" placeholder="Subject 3 Marks"><br>
```

```
<button onclick="addStudent()">Add Result</button>


<table>
  <thead>
    <tr>
      <th>Name</th>
      <th>Total</th>
      <th>Percentage</th>
      <th>Grade</th>
      <th>Status</th>
    </tr>
  </thead>
  <tbody id="resultTable"></tbody>
</table>


<script>
  function addStudent() {
    let name = document.getElementById("name").value;
    let marks = [
      Number(document.getElementById("sub1").value),
      Number(document.getElementById("sub2").value),
      Number(document.getElementById("sub3").value)
```

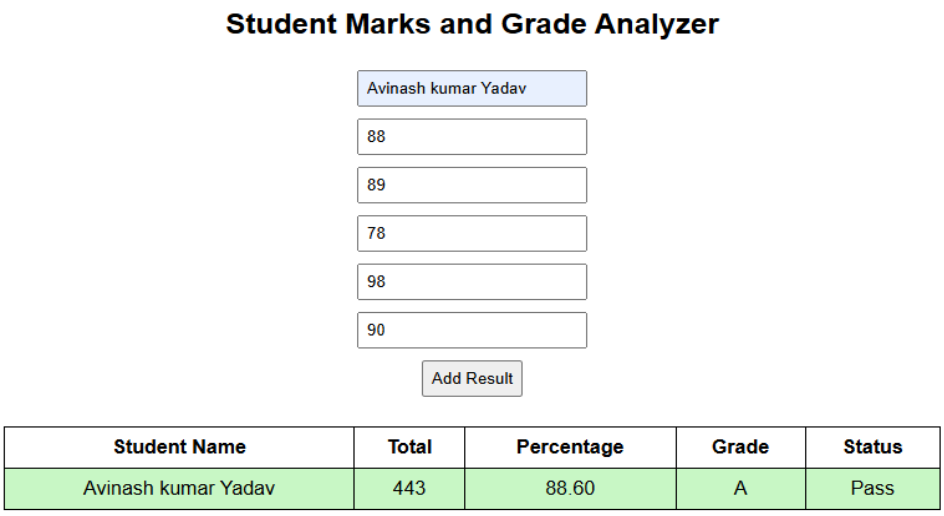
```
];

if (name === "" || marks.includes(NaN)) {
   alert("Please enter all details correctly");
   return;
}

let total = 0;
for (let i = 0; i < marks.length; i++) {
   total += marks[i];
}

let percentage = (total / (marks.length * 100)) * 100;

let grade;
if (percentage >= 80) grade = "A";
else if (percentage >= 60) grade = "B";
else if (percentage >= 40) grade = "C";
else grade = "F";

let status = percentage >= 40 ? "Pass" : "Fail";
```

```
        let student = {
            name: name,
            total: total,
            percentage: percentage.toFixed(2),
            grade: grade,
            status: status
        };

        let row = document.createElement("tr");
        row.className = status === "Pass" ? "pass" : "fail";

        row.innerHTML = `
            <td>${student.name}</td>
            <td>${student.total}</td>
            <td>${student.percentage}%</td>
            <td>${student.grade}</td>
            <td>${student.status}</td>
        `;

        document.getElementById("resultTable").appendChild(row);
    }
</script>
```

</body>

</html>

5. Output Screenshots



Conclusion
The Student Marks & Grade Analyzer successfully uses Vanilla JavaScript to store student details as objects, calculate academic results, and dynamically display them in a table with visual pass/fail indicators using JavaScript-controlled CSS.

Assignment 3: Dynamic To-Do List with Status Filter

1. Problem Understanding

The objective of this assignment is to build a Dynamic To-Do List application using Vanilla JavaScript.

The application allows users to add tasks, mark tasks as completed, and delete tasks.
It also provides filter options to view All, Completed, or Pending tasks and displays the total number of completed tasks dynamically.

No external libraries or frameworks are used.

2. List of JavaScript Concepts Used

- Arrays
- Objects
- Functions
- Conditional statements
- Loops (for, forEach)
- DOM Manipulation
    - document.getElementById()
    - createElement()
    - innerHTML
- Event handling (onclick, onchange)
- Dynamic CSS manipulation using JavaScript

3. Flow Diagram / Logic Steps

1. User enters a task
2. Task is stored as an object in an array
3. Task list is displayed dynamically
4. User can mark a task as completed

5. User can delete a task
6. Filter option is selected (All / Completed / Pending)
7. Tasks are displayed based on filter
8. Completed task count is updated

4. Source Code

HTML + JavaScript Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Dynamic To-Do List</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 40px;
    }
    input, button, select {
      padding: 6px;
      margin: 5px;
    }
```

```
        ul {
            list-style-type: none;
            padding: 0;
            width: 40%;
            margin: auto;
        }
        li {
            padding: 8px;
            margin: 5px 0;
            background: #f0f0f0;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }
        .completed {
            text-decoration: line-through;
            background-color: #c8f7c5;
        }
    </style>
</head>
<body>
```

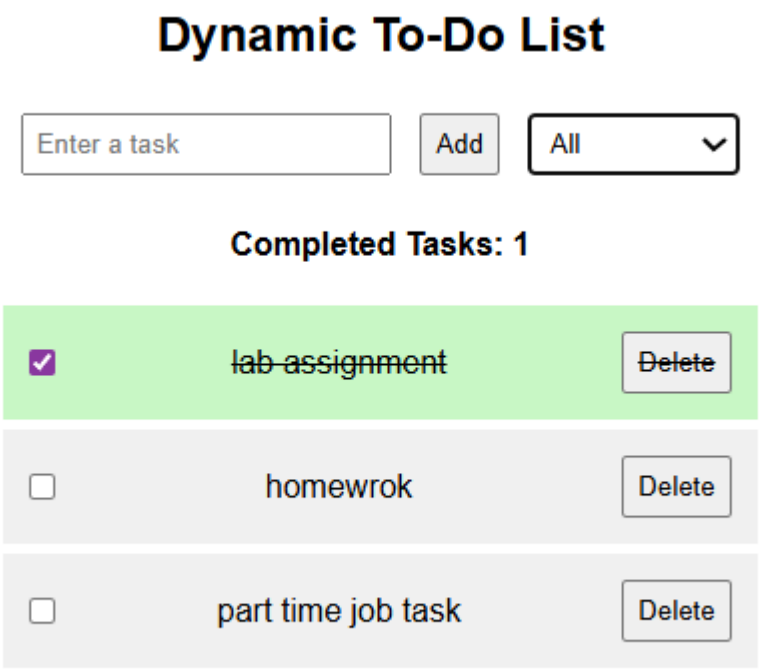
```
<h2>Dynamic To-Do List</h2>

<input type="text" id="taskInput" placeholder="Enter a task">
<button onclick="addTask()">Add</button><br>

<select id="filter" onchange="displayTasks()">
  <option value="all">All</option>
  <option value="completed">Completed</option>
  <option value="pending">Pending</option>
</select>

<h4>Completed Tasks: <span id="count">0</span></h4>

<ul id="taskList"></ul>

<script>
  let tasks = [];

  function addTask() {
    let text = document.getElementById("taskInput").value;

    if (text === "") {
```

```
        alert("Task cannot be empty");
        return;
    }

    tasks.push({ text: text, completed: false });
    document.getElementById("taskInput").value = "";
    displayTasks();
}

function displayTasks() {
    let list = document.getElementById("taskList");
    let filter = document.getElementById("filter").value;
    list.innerHTML = "";

    let completedCount = 0;

    tasks.forEach((task, index) => {
        if (
            filter === "all" ||
            (filter === "completed" && task.completed) ||
            (filter === "pending" && !task.completed)
        ) {
```

```
let li = document.createElement("li");
li.className = task.completed ? "completed" : "";

let checkbox = document.createElement("input");
checkbox.type = "checkbox";
checkbox.checked = task.completed;
checkbox.onchange = function () {
    tasks[index].completed = this.checked;
    displayTasks();
};

let span = document.createElement("span");
span.innerText = task.text;

let delBtn = document.createElement("button");
delBtn.innerText = "Delete";
delBtn.onclick = function () {
    tasks.splice(index, 1);
    displayTasks();
};

li.appendChild(checkbox);
```

```
            li.appendChild(span);
            li.appendChild(delBtn);
            list.appendChild(li);
        }

        if (task.completed) completedCount++;
    });

    document.getElementById("count").innerText = completedCount;
  }
</script>

</body>
</html>
```

5. Output Screenshots



Conclusion

The Dynamic To-Do List application successfully demonstrates the use of Vanilla JavaScript to manage tasks dynamically, apply filters, update UI elements, and track completed tasks without relying on any external libraries or frameworks.

Assignment 4: Number Guessing Game with Score Tracking

1. Problem Understanding

The objective of this assignment is to develop a **Number Guessing Game** using **Vanilla JavaScript**.
In this game, the system randomly generates a number between **1 and 100**.
The user repeatedly guesses the number, and the system provides feedback indicating whether the guess is **too high**, **too low**, or **correct**.
A **score tracking mechanism** is included, where the score decreases for every incorrect attempt.

2. List of JavaScript Concepts Used

- Variables (let, const)
- Random number generation (Math.random())

- Functions
- Conditional statements (if, else if, else)
- DOM Manipulation
    - document.getElementById()
    - innerText
- Event handling (onclick)
- Basic arithmetic operations

3. Flow Diagram / Logic Steps

1. System generates a random number between 1 and 100
2. Initial score is set
3. User enters a guess
4. Input validation is performed
5. Guess is compared with the random number
6. Feedback is displayed (High / Low / Correct)
7. Score decreases if the guess is wrong
8. Game ends when the correct number is guessed or score reaches zero

4. Source Code

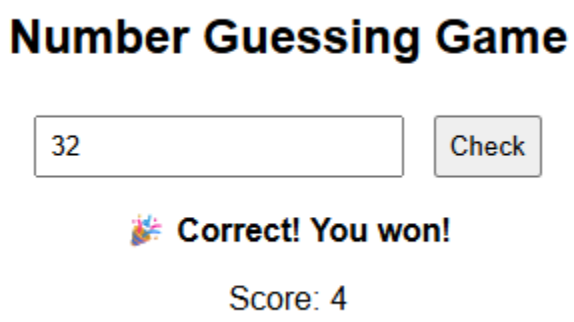
HTML + JavaScript Code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Number Guessing Game</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      text-align: center;
      margin-top: 50px;
    }
    input, button {
      padding: 6px;
      margin: 5px;
    }
    #message {
      font-weight: bold;
      margin-top: 10px;
    }
  </style>
</head>
```

```
<body>

<h2>Number Guessing Game</h2>

<p>Guess a number between <b>1 and 100</b></p>

<input type="number" id="guess" placeholder="Enter your guess">
<button onclick="checkGuess()">Check</button>

<p id="message"></p>
<p>Score: <span id="score">10</span></p>

<script>
  let randomNumber = Math.floor(Math.random() * 100) + 1;
  let score = 10;

  function checkGuess() {
    let userGuess = Number(document.getElementById("guess").value);

    if (userGuess < 1 || userGuess > 100) {
      document.getElementById("message").innerText = "Enter a number between 1 and
100";
      return;
    }

    if (userGuess === randomNumber) {
      document.getElementById("message").innerText = "🎉 Correct! You won!";
    }
    else if (userGuess > randomNumber) {
      document.getElementById("message").innerText = "Too High!";
      score--;
    }
    else {
      document.getElementById("message").innerText = "Too Low!";
      score--;
    }

    document.getElementById("score").innerText = score;

    if (score === 0 && userGuess !== randomNumber) {
      document.getElementById("message").innerText =
```

```
            "Game Over! The number was " + randomNumber;
        }
    }
</script>

</body>
</html>
```

5. Output Screenshots



Conclusion

The Number Guessing Game successfully demonstrates the use of **Vanilla JavaScript** to implement random number generation, conditional logic, score tracking, and dynamic user feedback without using any external libraries or frameworks

Assignment 5 : Interactive Quiz Application

1. Problem Understanding

The objective is to develop an Interactive Quiz Application using Vanilla JavaScript.
The system should:
Store questions as objects
Display one question at a time
Allow the user to select an answer
Calculate score and display it at the end
No external libraries or frameworks are used. The focus is on logic, DOM manipulation, and dynamic interaction.

2. List of JavaScript Concepts Used

- Objects and Arrays
- Functions
- Conditional statements (if, else)
- Loops (forEach)
- DOM Manipulation (getElementById, innerHTML)
- Event Handling (onclick)
- Variables (let, const)

3. Flow Diagram / Logic Steps

1. Define an array of question objects
2. Display the **first question** and its options
3. User selects an answer and clicks **Next**
4. Check if the selected answer is correct
5. Update the **score**
6. Move to the **next question**
7. Repeat until all questions are answered
8. Display **final score**

4. Source Code (Vanilla JavaScript Only)

HTML + JavaScript Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Interactive Quiz App</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin-top: 40px;
        }
        .option {
            display: block;
            margin: 10px 0;
        }
        button {
            padding: 8px 12px;
            margin-top: 10px;
        }
    </style>
</head>
<body>
```

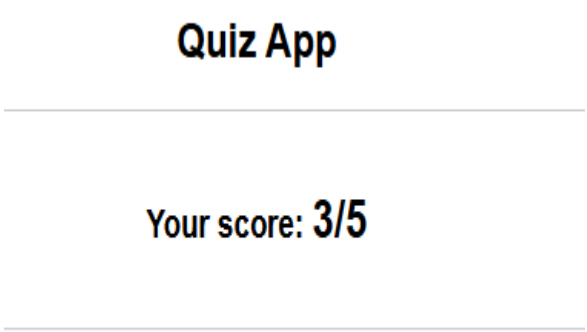
```
<h2>Interactive Quiz App</h2>

<div id="quizContainer">
  <p id="question"></p>
  <div id="options"></div>
  <button onclick="nextQuestion()">Next</button>
</div>

<div id="result" style="display:none;">
  <h3>Your Score: <span id="score"></span></h3>
</div>

<script>
  const questions = [
    {
      question: "What is the capital of Nepal?",
      options: ["Kathmandu", "Pokhara", "Lalitpur", "Bhaktapur"],
      answer: "Kathmandu"
    },
    {
      question: "Which language is used for web apps?",
```

```
        options: ["Python", "JavaScript", "C++", "Java"],
        answer: "JavaScript"
    },
    {
        question: "Who is known as the father of computers?",
        options: ["Alan Turing", "Charles Babbage", "Steve Jobs", "Bill Gates"],
        answer: "Charles Babbage"
    }
];

let currentQuestion = 0;
let score = 0;

function loadQuestion() {
    const q = questions[currentQuestion];
    document.getElementById("question").innerText = q.question;
    const optionsDiv = document.getElementById("options");
    optionsDiv.innerHTML = "";

    q.options.forEach(option => {
        const radio = document.createElement("input");
        radio.type = "radio";
```

```
        radio.name = "option";
        radio.value = option;

        const label = document.createElement("label");
        label.className = "option";
        label.appendChild(radio);
        label.appendChild(document.createTextNode(option));

        optionsDiv.appendChild(label);
    });
}

function nextQuestion() {
    const selectedOption =
                        document.querySelector('input[name="option"]:checked');

    if (!selectedOption) {
        alert("Please select an answer");
        return;
    }

    if (selectedOption.value === questions[currentQuestion].answer) {
```

```
            score++;
        }

        currentQuestion++;

        if (currentQuestion < questions.length) {
            loadQuestion();
        } else {
            document.getElementById("quizContainer").style.display = "none";
            document.getElementById("result").style.display = "block";
            document.getElementById("score").innerText = score + " / " +
                                questions.length;
        }
    }

    loadQuestion();
</script>

</body>
</html>
```

5. Output Screenshots



Conclusion

The Interactive Quiz Application demonstrates the use of Vanilla JavaScript for object storage, DOM manipulation, user interaction, and score calculation. It efficiently shows one question at a time and provides a dynamic, interactive experience without any external libraries.