NITTE (Deemed to be University) | NMAM INSTITUTE OF TECHNOLOGY

LAB MANUAL

Course: FRONT END WEB DEVELOPMENT

Course Code: CS1102-1

SEM: III

Department of Computer Science & Engineering

Year: 2024-25

Department of Computer Science & Engineering

Vision

To be a center of excellence in Computer science & Engineering education and research, empower the lives of individuals to fulfill their academic excellence, professional passions, and partnership for community development.

Mission

- To impart knowledge through the state-of-the-art concepts and technologies in Computer Science and Engineering.
- To inculcate values of professional ethics, leadership qualities and lifelong learning.
- To prepare professionals for employment in industry, research, higher education, community partnerships, and entrepreneurship to benefit the society.

Program Educational Objectives (PEOs)

- Graduates will be capable of practicing principles of Computer Science & Engineering, Mathematics and Scientific investigation to solve the problems that are appropriate to the discipline.
- Graduates will be able to contribute to their profession and society.
- Graduates will be employed in computing profession or engaged in learning to pursue higher education.

Program Specific Outcomes (PSOs)

- Foundations of Mathematical Concepts: Apply the knowledge of engineering science and mathematics in solving problems that are appropriate to the discipline.
- Foundations of Computing: Apply the knowledge of computing both hardware and software aspects to the solution of real-world engineering problems in the discipline.
- Foundations of Software Design & Development: Design & develop algorithms, programs, and projects using various software and modern tools appropriate to the software industry or Research & Development activities in the discipline.

Program Outcomes (POs)

PO1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and Analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

P04: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

P09: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Content

Subject Code : CS1102-1	**Hrs/Week : 1:1:3**
Hours/Week: 5	**CIE: 50**
Teaching Hours: 15+39	**SEE: 50**
Exam Hours: 3	**Credits: 03**

UNIT – I	
HTML and CSS, Version Control Introduction to Front-End, Structure of HTML and tags in HTML, Inline and Block Elements, Tags in HTML, Creating Your First Page, Introduction to Forms. Introduction to CSS, CSS Syntax, CSS Properties, Flexbox and CSS, CSS Grid and Responsive CSS. Why Version Control? Overview of Version Control System, Git and GitHub, Making Your Project Remote, Accessing Previous Commits, Branching, Working with branches, Managing Conflicts, Collaboration, Cloning. Program List: 1. Design a web application to implement different tags in HTML. 2. Develop a web form using CSS properties. 3. Design a web login page using responsive CSS concept. Build a simple portfolio website and demonstrate uploading and making project remote inGitHub.	15Hrs
UNIT – II	
Introduction to JavaScript Introduction to Basic JavaScript, Variables in JavaScript, Comparison and Assignment Operators, Conditional Statements, Loops, Conditionals and Error Handling, Objects, Object prototype & properties,Arrays, I/O Model, Document Object Model, ID and Class Selectors, Tag Selectors in JavaScript, Creating Tag Elements in JavaScript, Styling with JavaScript, Events in JavaScript, Variables and Loops in DOM Manipulation, Functions in JavaScript, Event Listeners in JavaScript. Program List: 1. Write a JavaScript code to demonstrate conditional statement and loops. 2. Design a JavaScript code to illustrate functions. 3. Design a JavaScript code to illustrate DOM manipulation. 4. Design a JavaScript code to illustrate JS Events.	15 Hrs

UNIT – III	
Introduction to AJAX and Node.js Introducing AJAX, How AJAX Works, A Simple AJAX Example, Create an XMLHttpRequest GET and POST, AJAX Response Formats and AJAX Applications. Introduction of Node.js, Node.js Modules: Built-in Modules, Include Modules and Create Your Own Modules, Node.js HTTP Module and NPM, Creating Web Server: Sending Requests, Handling HTTP requests Node.js Events: Event Emitter class, Inheriting Events and Returning event emitter. Program List: 1. Design an app for Sending Data to Server using XMLHttpRequest. 2. Write an example code to explain AJAX call. 3. Design a Node.js code to illustrate Modules. 4. Design a Node.js code to illustrate Events.	09Hrs

Textbooks:

1. Lisa Lopuck , Web Design For Dummies (For Dummies Series), 2012
2. 1. M. Deitel, P.J. Deitel, A. B. Goldberg," Internet & World Wide Web: How to Program, 4e
   Paperback – 1 January 2009.
3. 2. Chris Bates,"Web Programming Building Internet Applications", Third Edition, Wiley India,
   2006

E Books / MOOCs/ NPTEL

1. https://www.cs.uct.ac.za/mit_notes/web_programming.html
2. http://www.multitech.ac.ug/uploads/IntroductiontoWebProgramming.pdf
3. https://www.w3schools.com/bootstrap/
4. https://www.w3schools.com/nodejs/
5. https://www.tutorialspoint.com/nodejs/index.htm
6. http://nptel.ac.in/courses/106106156/2
7. https://www.coursera.org/learn/web-development

Course outcomes

After going through this course, the students will be able to:

Course Outcomes	Pos
CO1: Apply HTML and CSS for creating interactive, visually appealing, and responsive webpages and forms.	PO1, PO2, PO12
CO2:Utilize Git and GitHub for efficient project management, collaboration, and change trackingin website development	PO1, PO2, PO3, PO12
CO3: Demonstrate the ability to write, debug, and execute JavaScript code using variables, operators, conditionals, and loops.	PO1, PO2, PO3, PO12
CO4: Implement event handling using event listeners and user interactions.	PO1, PO2, PO3, PO12
CO5: Apply AJAX and Node.js Techniques to Develop Dynamic Web Applications	PO1, PO2, PO3, PO12

Table: Mapping Levels of COs to POs/PSOs

Course Outcomes Mapping with Program Outcomes & PSO															
Course Outcome	**Program Outcomes (POs)**												**PSOs**		
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
CS1102-1.1	3	2										2	3		
CS1102-1.2	3	2	1									2	3		
CS1102-1.3	3	2	1									2	3		
CS1102-1.4	3	2	1									2	3		
CS1102-1.5	3	2	2									2	3		

1: Low 2: Medium 3: High

FRONT END WEB DEVELOPMENT LAB PROGRAMS

Course Code: CS1102-1 **Course Type PCC**
Teaching Hours/Week (L: T: P) 1:1:3 **Credits 03**
Total Teaching Hours 15+39 **CIE + SEE Marks 50+50**
Prerequisite NIL

Teaching Department: Computer Science and Engineering

Course Objectives:

- Be able to Develop Proficiency in Front-End Web Development.
- Be able to effectively utilize Git and GitHub for managing project versions.
- Be familiar with client-side JavaScript application development.
- Be able to Develop Proficiency in Event Handling and DOM Manipulation.
- Be able to understand the principles behind AJAX and its operation.
- Be able to effectively utilize Node.js for server-side development.

List of LAB Programs

Program No	Description	CO Mapped
1a	Design a website for your institution with a landing page containing the Institution Logo, Name, Affiliation, and Address in the header. Then a Menu Bar containing Menus like Home, Departments, Resources Contact US, etc. The footer should contain links to different social media sites and copyright information. It should contain at least two departments. Use HTML and CSS to build this website. Also, make the website responsive.	CS1102-1.1
1b	Develop a portfolio website containing your photo and Personal Profile. Demonstrate the following using GitHub I. How to create a GitHub Repository Ii. How to clone the repository to the local drive Iii. How to push and commit the code Iv. Make the website or webpage in the repository to be remotely accessible.	CS1102-1.2
2a	Design a Webpage for an E-commerce website that provides the user to do the following I. Create a sign-up page for the new user that accepts necessary user details Ii. Create a sign-in page for the existing users Note: Use HTML and CSS to build the above website, and ensure the website is responsive.	CS1102-1.1
2b	Design a web page that demonstrates CSS animation to create the following I. Create a Scrollable text that scrolls from left to right at a given interval Ii. Create a c that blinks at a regular interval	CS1102-1.1

3a	Implement operations to manage student records using JavaScript objects and arrays. Include error handling to manage invalid data entries and failed operations.	CS1102-1.3
3b	Develop a traffic light simulator using JavaScript. Implement the logic for the changing lights using conditional statements.	CS1102-1.3
4a	Write JavaScript code to create a dynamic table that allows users to add and remove rows.	CS1102-1.4
4b	Design a form validation system using JavaScript that provides real-time feedback to users.	CS1102-1.4
5a	Create a simple webpage that allows users to enter a city name. Upon clicking a button, make an AJAX call to a weather API, fetch weather data for the specified city, and display the temperature and weather conditions on the page.	CS1102-1.5
5b	Design a webpage with two dropdowns, where the selection in the first dropdown dynamically populates the options in the second dropdown using AJAX calls to fetch related data from the server.	CS1102-1.5
6a	Develop a Node.js to explore different types of Node.js modules and their usage. I. Create a module that exports functions to perform basic arithmetic operations. II. Import and use this module in a main Node.js script to perform calculations. III. Experiment with built-in modules like fs (File System) to read/write files. IV. Test the module functionality by invoking exported functions with different inputs.	CS1102-1.5
6b	Develop a Node.js to explore event-driven programming in Node.js using the Event Emitter class. I. Create an event emitter that triggers custom events based on user actions or system events. II. Implement event listeners to handle these events and perform specific actions	CS1102-1.5

Lab Evaluation Scheme

Rubrics for Continuous Evaluation in Regular Labs:

Rubrics For Evaluation	Distribution Of Marks
Execution	5
Quality of output, Viva	5
Record	10
TOTAL	**20**

Marks Distribution for Lab MSE:

Evaluation	Distribution Of Marks
Write up	5+5
Execution of Program	10+10
TOTAL	**30**

Marks Distribution for Final Lab CIE:

Lab CIE	Distribution Of Marks
Average of ten marks of continuous evaluation	10
Average of ten Marks of Record	10
Lab MSE	30
TOTAL	**50**

LAB PROGRAMS

1a. Design a website for your institution with a landing page containing the Institution Logo, Name, Affiliation, and Address in the header. Then a Menu Bar containing Menus like Home, Departments, Resources Contact US, etc. The footer should contain links to different social media sites and copyright information. It should contain at least two departments. Use HTML and CSS to build this website. Also, make the website responsive.

SAMPLE CODE

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="styles.css">
<title>Your Institution</title>
</head>
<body>
<header>
<div class="logo">
<img src=" " alt="Institution Logo">
</div>
<div class="header-content">
<h1>Institution Name</h1>
<p>Affiliation</p>
<p>Address</p>
</div>
</header>
<nav>
<ul class="menu">
<li><a href="#">Home</a></li>
<li class="submenu">
<a href=" ">Departments</a>
<ul class="submenu-content">
<li><a href=" ">Department of CSE</a></li>
```

```
<li><a href=" ">Department of ISE</a></li>
</ul>
</li>
<li><a href="#">Resources</a></li>
<li><a href="#">Contact Us</a></li>
</ul>
</nav>
<main>

ABOUT COLLEGE

</main>
<footer>
<div class="social-links">
<a href="#">Facebook</a>
<a href="#">Twitter</a>
<a href="#">Instagram</a>
</div>
<p>&copy; 2023 Your Institution. All rights reserved.</p>
</footer>
</body>
</html>
```

CSS FILE

```
/* Reset some default styles */
body, h1, h2, p, ul, li {
margin: 0;
padding: 0;
}
/* Global Styles */
body {
font-family: Arial, sans-serif;
line-height: 1.6;
}
```

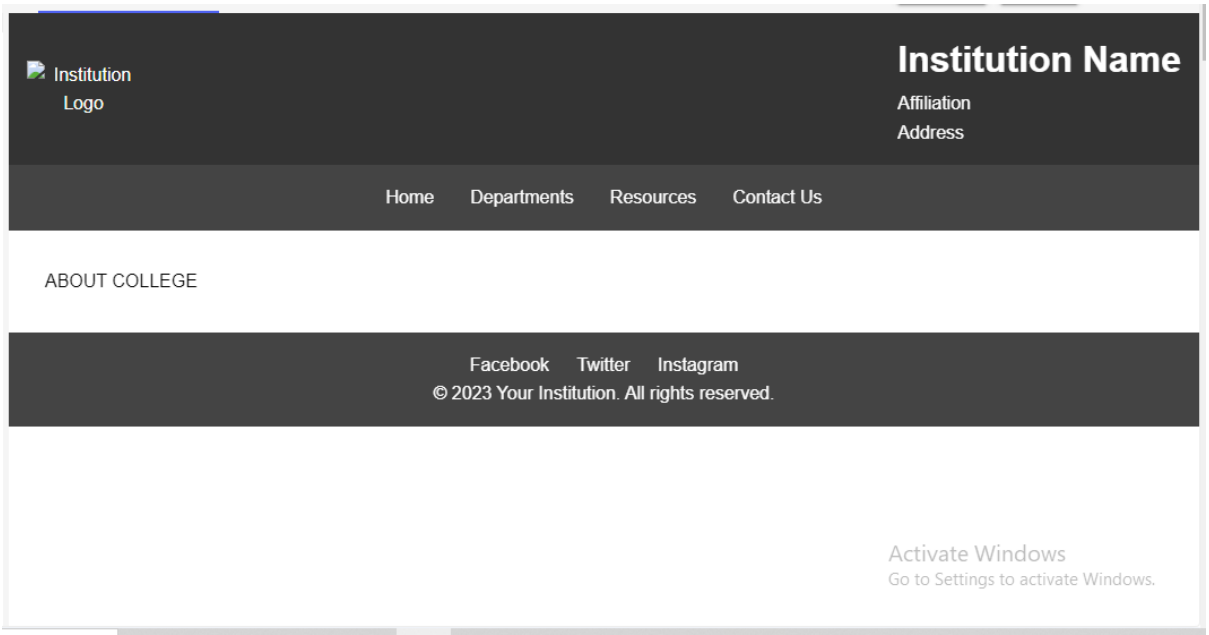
```
header {
background-color: #333;
color: #fff;
text-align: center;
padding: 1rem;
display: flex;
align-items: center;
justify-content: space-between;
}
.logo img {
max-width: 100px;
height: auto;
}
.header-content {
text-align: left;
}

nav {
background-color: #444;
}
.menu {
list-style: none;
display: flex;
justify-content: center;
padding: 1rem 0;
}
.menu li {
margin: 0 1rem;
position: relative;
}
.menu a {
color: #fff;
```

```
text-decoration: none;
transition: color 0.3s;
}
.menu a:hover {
color: #ff9900;
}
.submenu-content {
display: none;
position: absolute;
top: 100%;
left: 0;
background-color: #444;
z-index: 1;
}
.submenu:hover .submenu-content {
display: block;
}
main {
padding: 2rem;
}
.department {
margin-bottom: 2rem;
}

footer {
text-align: center;
padding: 1rem 0;
background-color: #444;
color: #fff;
}
.social-links a {
color: #fff;
```

```
text-decoration: none;
margin: 0 10px;
}
/* Media Queries */
@media screen and (max-width: 768px) {
header {
flex-direction: column;
}
.logo {
margin-bottom: 1rem;
}
}
```

Add create department page, Resources, and Contact Us

Expected Outcome



1b. Develop a portfolio website containing your photo and Personal Profile. Demonstrate the following using GitHub.

I. How to create a GitHub Repository
II. How to clone the repository to the local drive.
III. How to push and commit the code.
IV. Make the website or webpage in the repository to be remotely accessible.

Step 1: Create a GitHub Repository

Sign in or create an Account: If you don't have a GitHub account, go to GitHub and sign up.

Create a New Repository:

● Click on the "+" sign in the top right corner of the GitHub interface and select "New Repository."

● Fill in the repository name (e.g., "portfolio-website") followed by the github.io extension.

● Choose visibility (public or private) based on your preference.

● Optionally you can initialize the repository with a README file.

Step 2: Clone the Repository to the Local Drive

● Sign in to GitHub for Desktop and add the repository then Navigate to the directory where you want to clone the repository and finally clone it. In the local drive, a copy of the repository will be generated.

Step 3: Push and Commit the Code

Add Your Portfolio Files:

● Add your portfolio website files (HTML, CSS, images, etc.) to the cloned repository directory.

● Stage and Commit Changes:

Push Changes to GitHub:

After committing, push your changes to the GitHub repository using:

Step 4: Make the Website Remotely Accessible

GitHub Pages:

● Go to your GitHub repository on the GitHub website.

● Click on the "Settings" tab.

● Scroll down to the "GitHub Pages" section.

● Choose the main branch (or your preferred branch) as the source.

● Click "Save."

SAMPLE CODE

Index.html

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="styles.css">
<title>Your Portfolio</title>
</head>
<body>
<header>
<h1>Your Name</h1>
</header>

<main>
<section class="profile">
<div class="photo">
<img src="your-photo.jpg" alt="Your Photo">
</div>
<div class="bio">
<h2>About Me</h2>
<p>

Write a brief description of yourself here. Mention your interests, skills, and

experiences.
</p>
</div>
</section>
</main>
<footer>
```

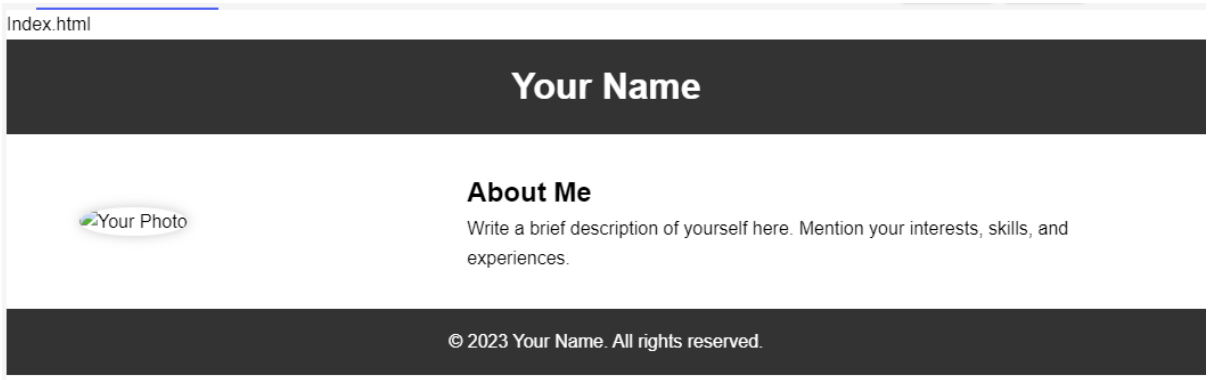
```
<p>&copy; 2023 Your Name. All rights reserved.</p>
</footer>
</body>
</html>
```

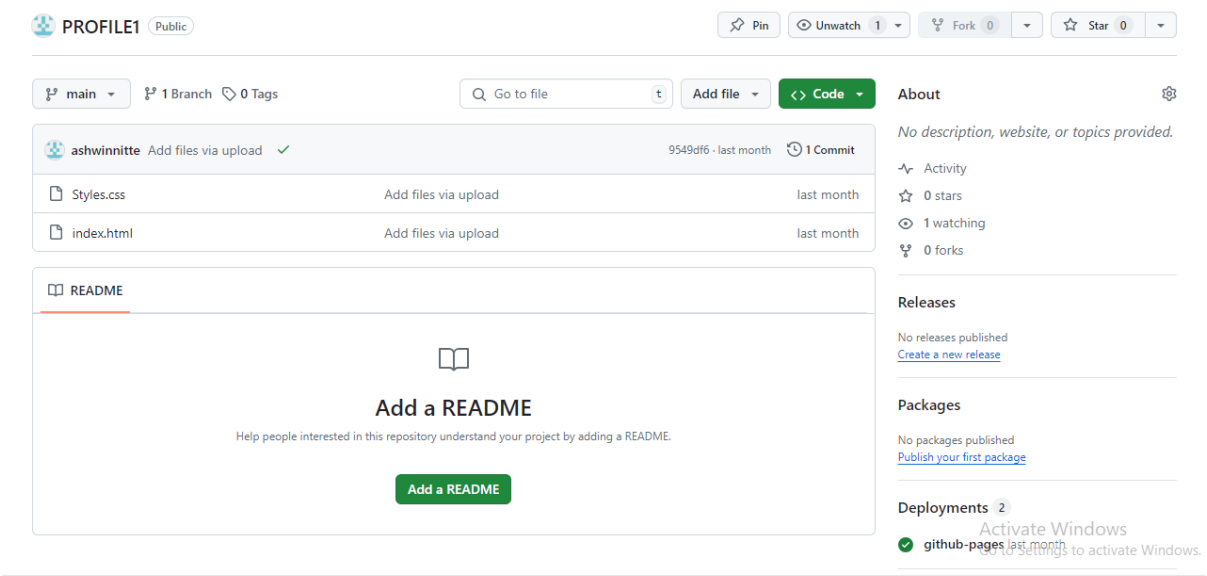
Styles.css

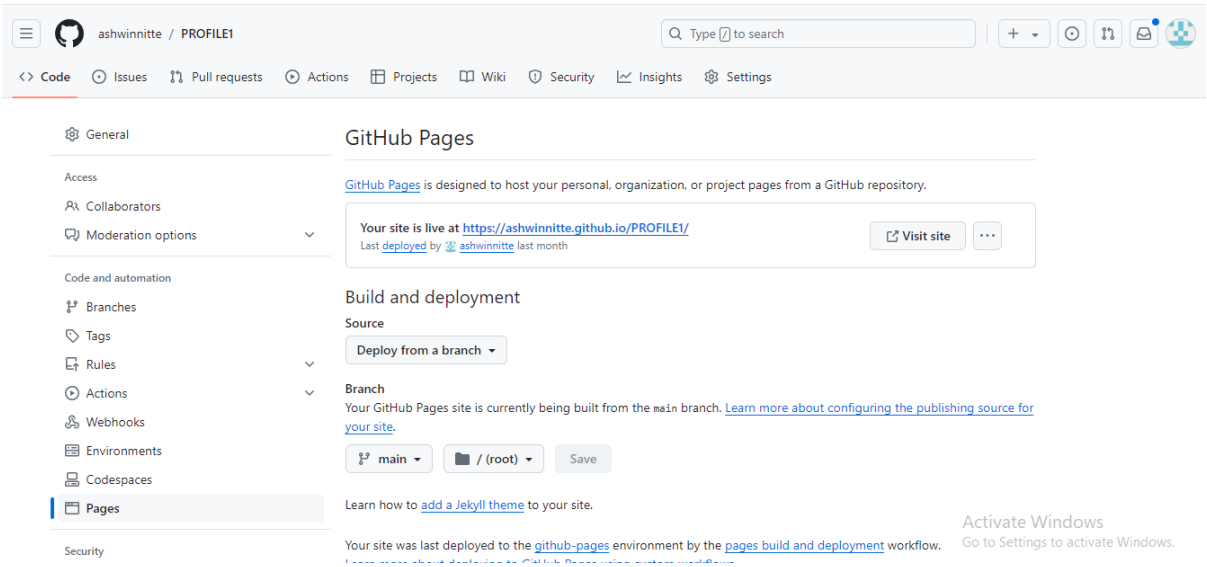
```
body, h1, h2, p {
margin: 0;
padding: 0;
}
/* Global Styles */
body {
font-family: Arial, sans-serif;
line-height: 1.6;
}
header {
background-color: #333;
color: #fff;
text-align: center;
padding: 1rem;
}
main {
padding: 2rem;
}
.profile {
display: flex;
align-items: center;
}
.photo {
flex: 1;
padding: 0 2rem;
}
```

```
.photo img {
max-width: 100%;
height: auto;
border-radius: 50%;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
.bio {
flex: 2;
}
footer {
text-align: center;
padding: 1rem 0;
background-color: #333;
color: #fff;
}
/* Media Queries */
@media screen and (max-width: 768px) {
.profile {
flex-direction: column;
}}
```

Expected Outcome




PROFILE1
Public
Pin
Unwatch 1
Fork 0
Star 0
main
1 Branch
0 Tags
Go to file
t
Add file
Code
About
No description, website, or topics provided.
ashwinnitte Add files via upload
9549df6 · last month
1 Commit
Styles.css
Add files via upload
last month
index.html
Add files via upload
last month
README
Add a README
Help people interested in this repository understand your project by adding a README.
Add a README
Activity
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Deployments 2
github-pages last month
Activate Windows
Go to Settings to activate Windows.


ashwinnitte / PROFILE1
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
GitHub Pages
GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.
Your site is live at https://ashwinnitte.github.io/PROFILE1/
Last deployed by ashwinnitte last month
Visit site
Build and deployment
Source
Deploy from a branch
Branch
Your GitHub Pages site is currently being built from the main branch. Learn more about configuring the publishing source for your site.
main
/ (root)
Save
Learn how to add a Jekyll theme to your site.
Your site was last deployed to the github-pages environment by the pages build and deployment workflow.
Activate Windows
Go to Settings to activate Windows.

2a. Design a Webpage for an E-commerce website that provides the user to do the following.

I. Create a sign-up page for the new user that accepts necessary user details.

II. Create a sign-in page for the existing users.

Note: Use HTML and CSS to build the above website, and ensure the website is responsive.

SAMPLE CODE

index.html (Main Page):

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="styles.css">
<title>E-commerce Website</title>
</head>
<body>
<header>
<h1>Welcome to Our E-commerce Store</h1>
</header>
<main>
<div class="cta">
<a href="signup.html">Sign Up</a>
<a href="signin.html">Sign In</a>
</div>
</main>
<footer>
<p>&copy; 2023 E-commerce Store. All rights reserved.</p>
</footer>
</body>
</html>
```

signup.html (Sign-Up Page):

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="styles.css">
<title>Sign Up</title>
</head>
<body>
<header>
<h1>Sign Up</h1>
</header>
<main>
<form class="signup-form">
<label for="name">Name:</label>
<input type="text" id="name" name="name" required>
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
<label for="password">Password:</label>
<input type="password" id="password" name="password" required>
<button type="submit">Sign Up</button>
</form>
</main>
<footer>
<p>&copy; 2023 E-commerce Store. All rights reserved.</p>
</footer>
</body>
</html>
```

signin.html (Sign-In Page):

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="styles.css">
<title>Sign In</title>
</head>
<body>
<header>
<h1>Sign In</h1>

</header>
<main>
<form class="signin-form">
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
<label for="password">Password:</label>
<input type="password" id="password" name="password" required>
<button type="submit">Sign In</button>
</form>
</main>
<footer>
<p>&copy; 2023 E-commerce Store. All rights reserved.</p>
</footer>
</body>
</html>
```

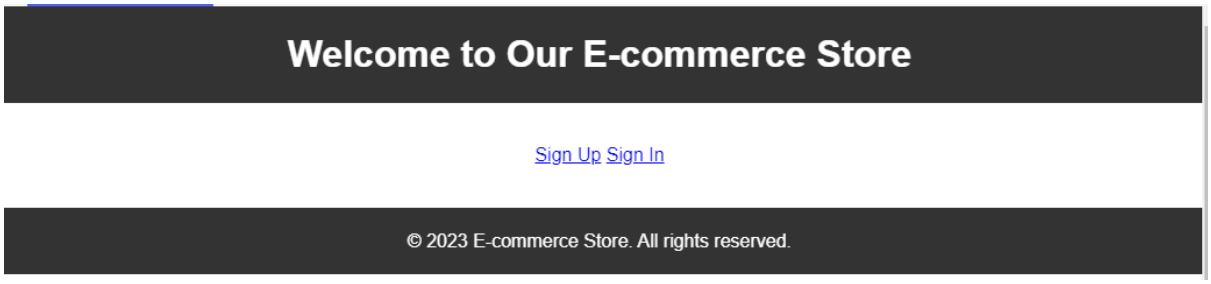
Css file

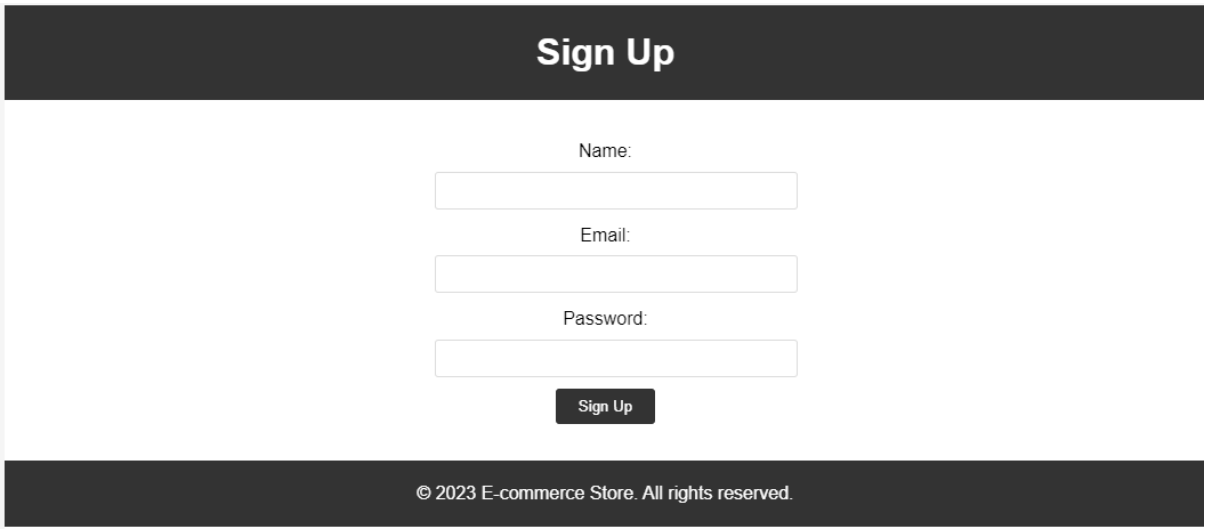
```
/* Reset some default styles */
body, h1, h2, p, label, input, button {
margin: 0;
```

```
padding: 0;
}
/* Global Styles */
body {
font-family: Arial, sans-serif;
line-height: 1.6;
}
header {
background-color: #333;
color: #fff;
text-align: center;
padding: 1rem;
}
main {

padding: 2rem;
text-align: center;
}
footer {
text-align: center;
padding: 1rem 0;
background-color: #333;
color: #fff;
}
/* Forms */
.signup-form, .signin-form {
max-width: 300px;
margin: 0 auto;
}
label {
display: block;
margin-bottom: 5px;
```

```
}
input {
width: 100%;
padding: 8px;
margin-bottom: 10px;
border: 1px solid #ddd;
border-radius: 3px;
}
button {
display: inline-block;
padding: 8px 20px;
background-color: #333;
color: #fff;
border: none;
border-radius: 3px;
cursor: pointer;
}
button:hover {
background-color: #555;
}
```

Expected Outcome




Sign Up
Name:
Email:
Password:
Sign Up
© 2023 E-commerce Store. All rights reserved.

2b. Design a web page that demonstrates CSS animation to create the following.

I. Create a Scrollable text that scrolls from left to right at a given interval.

II. Create a c that blinks at a regular interval.

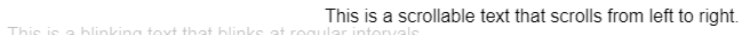
SAMPLE CODE

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
  body {
    display: flex;
    font-family: Arial, sans-serif;
    margin:5vh;
    padding: 10%;
    overflow: hidden;
  }

  .scrollable-text {
    position: absolute;
    white-space: nowrap;
    animation: scrollText 100s linear infinite;


  }

  .blinking-text {
    animation: blinkText 20s alternate infinite;

  }

  @keyframes scrollText {
    0% {
      transform: translateX(100%);
    }
    100% {
      transform: translateX(-100%);
    }
  }

  @keyframes blinkText {
    0%, 100% {
      opacity: 1;
    }
    50% {
      opacity: 0;
    }
  }
</style>
<title>CSS Animation Demo</title>
</head>
<body>
```

```
  <div class="scrollable-text">
    This is a scrollable text that scrolls from left to right.
  </div>
  <div class="blinking-text">
    <p>This is a blinking text that blinks at regular intervals.</p>
    <p>Show results in a different line.</p>
  </div>
</body>
</html>
```

Expected Outcome



This is a blinking text that blinks at regular intervals.

Show results in a different line.

UNIT 2

3a. Implement operations to manage student records using JavaScript objects and arrays. Include error handling to manage invalid data entries and failed operations.

SAMPLE 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>Student Records Management</title>

    <style>

        body {

            font-family: Arial, sans-serif;

            margin: 20px;

        }

        .error {

            color: red;

        }

    </style>

</head>

<body>

    <h1>Student Records Management</h1>

    <form id="studentForm">

        <label for="studentId">Student ID:</label>

        <input type="text" id="studentId" required>

        <br>
```

```
        <label for="studentName">Student Name:</label>
        <input type="text" id="studentName" required>
        <br>
        <label for="studentAge">Student Age:</label>
        <input type="number" id="studentAge" required>
        <br>
        <button type="submit">Add Student</button>
    </form>

    <h2>Student Records</h2>
    <div id="error" class="error"></div>
    <ul id="studentList"></ul>

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

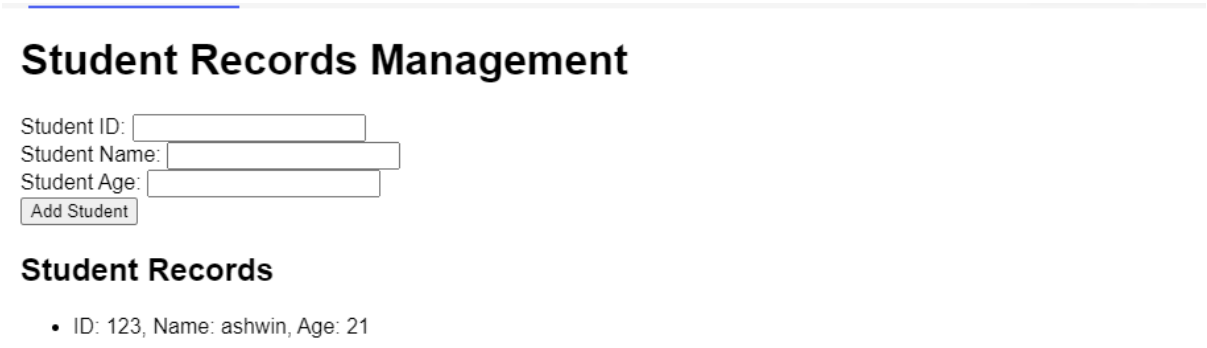
Scripit.js

```
const studentForm = document.getElementById('studentForm');
const studentList = document.getElementById('studentList');
const errorDiv = document.getElementById('error');
const students = [];
studentForm.addEventListener('submit', function(event) {
    event.preventDefault();

    const studentId = document.getElementById('studentId').value;
    const studentName = document.getElementById('studentName').value;
```

```
    const studentAge = document.getElementById('studentAge').value;

    try {
        addStudent(studentId, studentName, studentAge);
        displayStudents();
        errorDiv.textContent = '';
    } catch (error) {
        errorDiv.textContent = error.message;
    }

    studentForm.reset();
});

function addStudent(id, name, age) {
    if (!id || !name || !age) {
        throw new Error('All fields are required.');
    }

    if (isNaN(age) || age <= 0) {
        throw new Error('Age must be a positive number.');
    }

    const studentExists = students.some(student => student.id === id);

    if (studentExists) {
        throw new Error('Student ID already exists.');
    }
```

```
    const student = {
        id,
        name,
        age: parseInt(age, 10)
    };

    students.push(student);
}

function displayStudents() {
    studentList.innerHTML = '';

    students.forEach(student => {
        const li = document.createElement('li');
        li.textContent = `ID: ${student.id}, Name: ${student.name}, Age: ${student.age}`;
        studentList.appendChild(li);
    });
}
```

Expected Outcome



3b. Develop a traffic light simulator using JavaScript. Implement the logic for the changing lights using conditional statements.

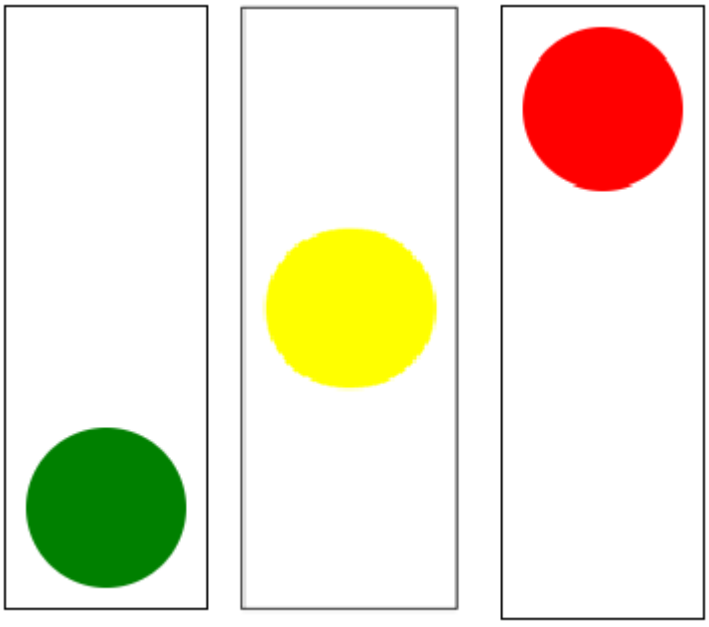
SAMPLE CODE

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Traffic Light Simulator</title>
<style>
  .traffic-light {
    width: 100px;
    height: 300px;
    border: 1px solid black;
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: space-between;
    margin-top: 50px;
  }
  .light {
    width: 80px;
    height: 80px;
    border-radius: 50%;
    margin: 10px;
  }
  .red {
    background-color: red;
  }
  .yellow {
    background-color: yellow;
  }
  .green {
    background-color: green;
  }
</style>
</head>
<body>
<div class="traffic-light">
  <div class="light red"></div>
  <div class="light yellow"></div>
  <div class="light green"></div>
</div>
<script>
  const redLight = document.querySelector('.red');
  const yellowLight = document.querySelector('.yellow');
  const greenLight = document.querySelector('.green');

  function changeLights() {
    setTimeout(() => {
      redLight.style.backgroundColor = 'red';
      yellowLight.style.backgroundColor = 'transparent';
      greenLight.style.backgroundColor = 'transparent';
```

```
      setTimeout(() => {
        redLight.style.backgroundColor = 'transparent';
        yellowLight.style.backgroundColor = 'yellow';
        greenLight.style.backgroundColor = 'transparent';

        setTimeout(() => {
          redLight.style.backgroundColor = 'transparent';
          yellowLight.style.backgroundColor = 'transparent';
          greenLight.style.backgroundColor = 'green';

          changeLights(); // Recursively call to keep the cycle going
        }, 2000); // Green light for 2 seconds
      }, 1000); // Yellow light for 1 second
    }, 2000); // Red light for 2 seconds
  }

  changeLights(); // Start the cycle
</script>
</body>
</html>
```

Expected Outcome



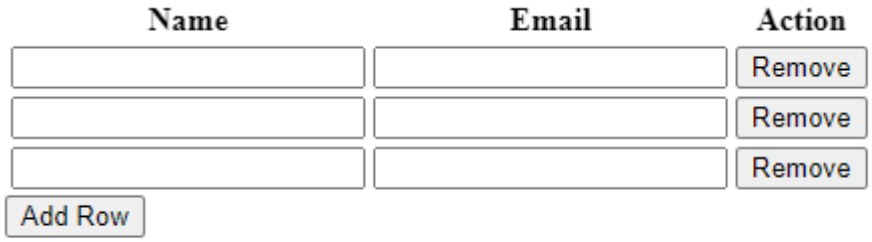
4a. Write JavaScript code to create a dynamic table that allows users to add and remove rows.

SAMPLE CODE

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Dynamic Table</title>
</head>
<body>
    <table id="dynamic-table">
        <thead>
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td><input type="text" name="name[]"></td>
                <td><input type="email" name="email[]"></td>
                <td><button type="button"
onclick="removeRow(this)">Remove</button></td>
            </tr>
        </tbody>
    </table>
    <button type="button" onclick="addRow()">Add Row</button>

    <script>
        function addRow() {
            var table = document.getElementById("dynamic-
table").getElementsByTagName('tbody')[0];
            var newRow = table.insertRow(table.rows.length);
            var cell1 = newRow.insertCell(0);
            var cell2 = newRow.insertCell(1);
            var cell3 = newRow.insertCell(2);
            cell1.innerHTML = '<input type="text" name="name[]">';
            cell2.innerHTML = '<input type="email" name="email[]">';
            cell3.innerHTML = '<button type="button"
onclick="removeRow(this)">Remove</button>';
        }

        function removeRow(button) {
            var row = button.parentNode.parentNode;
            row.parentNode.removeChild(row);
        }
    </script>
</body>
</html>
```

Expected Outcome



4b. Design a form validation system using JavaScript that provides real-time feedback to users.

SAMPLE 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>Simple Form Validation</title>
    <style>
        .error {
            color: red;
        }
        .valid {
            color: green;
        }
    </style>
</head>
<body>
    <form id="myForm">
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required>
        <span id="emailFeedback" class="error"></span>
        <br><br>

        <label for="password">Password:</label>
```

```
        <input type="password" id="password" name="password" required>
        <span id="passwordFeedback" class="error"></span>
        <br><br>

        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <span id="usernameFeedback" class="error"></span>
        <br><br>

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

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

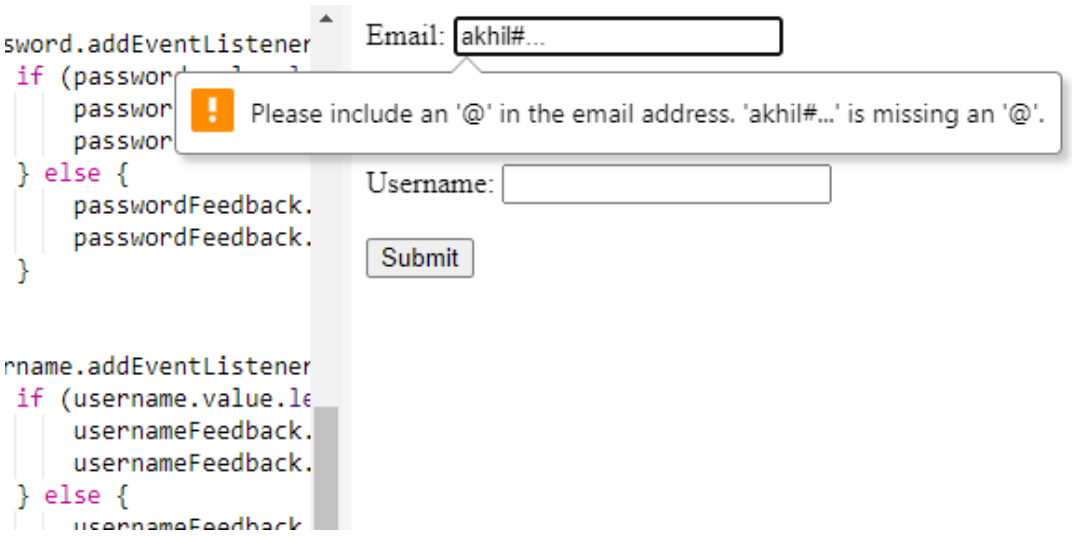
validation.js

```
document.addEventListener('DOMContentLoaded', () => {
    const form = document.getElementById('myForm');

    const email = document.getElementById('email');
    const password = document.getElementById('password');
    const username = document.getElementById('username');

    const emailFeedback = document.getElementById('emailFeedback');
    const passwordFeedback = document.getElementById('passwordFeedback');
```

```
    const                        usernameFeedback                        =
document.getElementById('usernameFeedback');



    email.addEventListener('input', () => {

        if (email.validity.valid) {

            emailFeedback.textContent = 'Valid email!';

            emailFeedback.className = 'valid';

        } else {

            emailFeedback.textContent = 'Please enter a valid email
address.';

            emailFeedback.className = 'error';

        }

    });



    password.addEventListener('input', () => {

        if (password.value.length >= 8) {

            passwordFeedback.textContent = 'Password is strong!';

            passwordFeedback.className = 'valid';

        } else {

            passwordFeedback.textContent = 'Password must be at least
8 characters long.';

            passwordFeedback.className = 'error';

        }

    });



    username.addEventListener('input', () => {

        if (username.value.length >= 3) {

            usernameFeedback.textContent = 'Username looks good!';
```

```
            usernameFeedback.className = 'valid';

        } else {

            usernameFeedback.textContent = 'Username must be at least
3 characters long.';

            usernameFeedback.className = 'error';

        }

    });



    form.addEventListener('submit', (event) => {

        if (!email.validity.valid || password.value.length < 8 ||
username.value.length < 3) {

            event.preventDefault();

            alert('Please fill out the form correctly before
submitting.');

        }

    });

});
```

Expected Outcome



UNIT 3

5a. Create a simple webpage that allows users to enter a city name. Upon clicking a button, make an AJAX call to a weather API, fetch weather data for the specified city, and display the temperature and weather conditions on the page.

Sampel code

Index.html

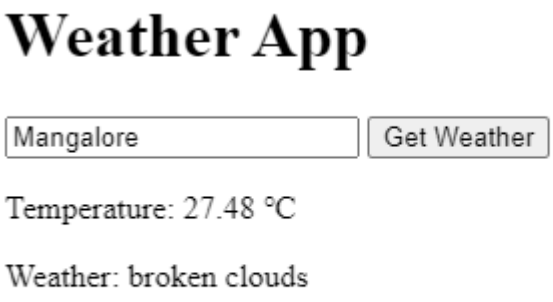
```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Weather App</title>
</head>
<body>
 <div id="app">
 <h1>Weather App</h1>
 <input type="text" id="cityInput" placeholder="Enter city name">
 <button id="fetchWeather">Get Weather</button>
 <div id="weatherResult">
 <!-- Weather data will be displayed here -->
 </div>
 </div>
 <script src="script.js"></script>
</body>
</html>
```

Script.js

```
document.addEventListener("DOMContentLoaded", function () {
 const apiKey = "b1693ed7ad181ad79817cc99d0523aa3"; // Replace with
your OpenWeatherMap API key
 const fetchWeatherButton = document.getElementById("fetchWeather");
 const cityInput = document.getElementById("cityInput");
 const weatherResult = document.getElementById("weatherResult");

 fetchWeatherButton.addEventListener("click", function () {
 const city = cityInput.value;
 if (city.trim() === "") {
 alert("Please enter a city name.");
 return;
 }

 const apiUrl =
`https://api.openweathermap.org/data/2.5/weather?q=${city}&appid=${apiK
ey}&units=metric`;

 fetch(apiUrl)
 .then((response) => response.json())
 .then((data) => {
 const temperature = data.main.temp;
 const weatherDescription = data.weather[0].description;
```

```
const weatherOutput = `
<p>Temperature: ${temperature} &#8451;</p>
<p>Weather: ${weatherDescription}</p>
`;

weatherResult.innerHTML = weatherOutput;
})
.catch((error) => {
console.error("Error fetching weather data:", error);
weatherResult.innerHTML = "Weather data not available.";
});
});
});
```

Expected Outcome



5b Design a webpage with two dropdowns, where the selection in the first dropdown dynamically populates the options in the second dropdown using AJAX calls to fetch related data from the server.

dropdown using AJAX calls to fetch related data from the server. get-cities.php

use winscp server

```
<?php
$country = $_GET["country"];
// Simulate fetching cities based on the selected country
if ($country === "usa") {
$cities = array("New York", "Los Angeles", "Chicago");
} elseif ($country === "canada") {
$cities = array("Toronto", "Vancouver", "Montreal");
} else {
$cities = array();
}
echo json_encode($cities);
?>
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Dynamic Dropdowns</title>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="script.js"></script>
</head>
<body>
```

```
<div class="container">

<h1>Dynamic Dropdowns Example</h1>
<label for="country">Select Country:</label>
<select id="country" name="country">
<option value="">Select</option>
<option value="usa">USA</option>
<option value="canada">Canada</option>
</select>

<label for="city">Select City:</label>
<select id="city" name="city">
<option value="">Select a country first</option>
</select>
</div>
</body>
</html>

script.js
$(document).ready(function () {
$("#country").change(function () {
const selectedCountry = $(this).val();

if (selectedCountry !== "") {
// Make an AJAX call to fetch cities based on the selected country
$.ajax({
url: `get-cities.php?country=${selectedCountry}`, type: "GET",
dataType: "json", success: function (data) {
const cityDropdown = $("#city");
cityDropdown.empty();
if (data.length > 0) {
cityDropdown.append("<option value=''>Select</option>");
```

```
$.each(data, function (index, city) {
cityDropdown.append(
$("<option></option>").attr("value", city).text(city)
);
});
} else {
cityDropdown.append("<option value=''>No cities available</option>");
}
}, error: function () {
console.error("Error fetching cities.");
}, });
} else {
// Reset city dropdown if no country is selected
$("#city").html("<option value=''>Select a country first</option>");
}
});
})
```

Expected Outcome

6a Develop a Node.js to Explore different types of Node.js modules and their usage.

I) Create a module that exports functions to perform basic arithmetic operations.

ii) Import and use this module in a main Node.js script to perform calculations.

iii) Experiment with built-in modules like fs (File System) to read/write files.

iv) Test the module functionality by invoking exported functions with different inputs.

Step 1: Create a Module for Arithmetic Operations

Create a file named arithmetic.js with the following content:

```
// arithmetic.js

function add(a, b) {
    return a + b;
}
function subtract(a, b) {
    return a - b;
}
function multiply(a, b) {
    return a * b;
}
function divide(a, b) {
    if (b === 0) {
        throw new Error('Division by zero is not allowed');
    }
    return a / b;
}
module.exports = {
    add,
    subtract,
    multiply,
```

```
    divide
};
```

Step 2: Create the Main Script to Use the Module

Create a file named main.js with the following content:

// main.js

```
const arithmetic = require('./arithmetic');
const fs = require('fs');

// Perform arithmetic operations
const a = 10;
const b = 5;

const sum = arithmetic.add(a, b);
const difference = arithmetic.subtract(a, b);
const product = arithmetic.multiply(a, b);
const quotient = arithmetic.divide(a, b);

// Output the results
console.log(`Sum: ${sum}`);
console.log(`Difference: ${difference}`);
console.log(`Product: ${product}`);
console.log(`Quotient: ${quotient}`);

// Write the results to a file
const results = `
Sum: ${sum}
Difference: ${difference}
Product: ${product}
Quotient: ${quotient}
```

```
`;

fs.writeFile('results.txt', results, (err) => {
    if (err) {
        console.error('Error writing to file', err);
    } else {
        console.log('Results written to results.txt');
    }
});

// Read the results from the file and display them
fs.readFile('results.txt', 'utf8', (err, data) => {
    if (err) {
        console.error('Error reading from file', err);
    } else {
        console.log('Content of results.txt:');
        console.log(data);
    }
});
```

Step 3: Test the Module Functionality

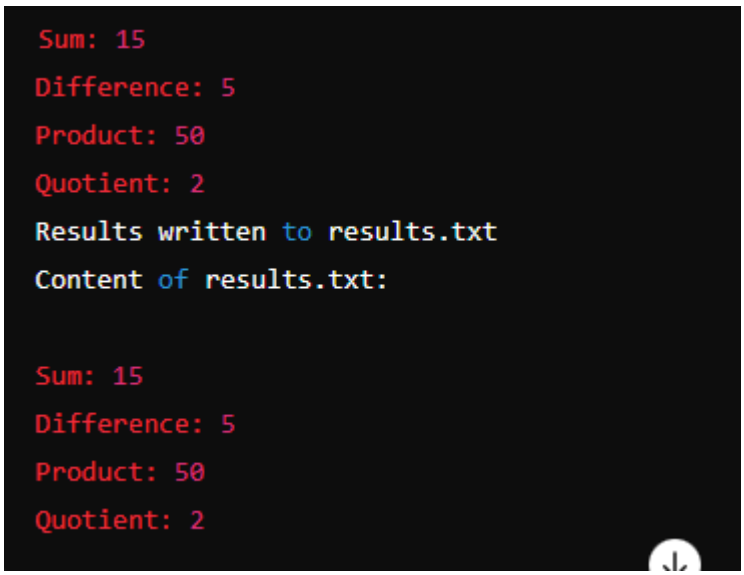
To test the module functionality, run the main.js script using Node.js:

```
node main.js
```

This script will:

1. Import the arithmetic module and the built-in `fs` module.
2. Perform basic arithmetic operations using the imported functions.
3. Output the results to the console.
4. Write the results to a file named `results.txt`.
5. Read the results from `results.txt` and display them on the console.

Expected Outcome



results.txt

Sum: 15

Difference: 5

Product: 50

Quotient: 2

6b Develop a Node.js to explore event-driven programming in Node.js using the Event Emitter class.

i) Create an event emitter that triggers custom events based on user actions or system events.

ii) Implement event listeners to handle these events and perform specific actions

Step 1: Setting Up the Project

Initialize a Node.js project:

```
npm init -y
```

Step 2: Implementing Event Emitter and Event Listeners

In the **app.js** file, add the following code:

```
// Import the 'events' module
const EventEmitter = require('events');

// Create an instance of the EventEmitter class
const eventEmitter = new EventEmitter();

// Define a custom event and its listener
eventEmitter.on('greet', (name) => {
  console.log(`Hello, ${name}!`);
});

eventEmitter.on('farewell', (name) => {
  console.log(`Goodbye, ${name}!`);
});
```

```
// Simulate user actions or system events
const userAction = (action, name) => {
  eventEmitter.emit(action, name);
};

// Trigger events based on user actions
userAction('greet', 'Alice');
userAction('farewell', 'Bob');
```

Step 3: Using the Event Emitter

Create a main.js file to use the eventEmitterExample.js module:

```
// main.js

const simulateUserActions = require('./eventEmitterExample');

// Call the function to simulate user actions
simulateUserActions();
```

Step 4: Testing the Event Emitter

Run the main.js script to test the event emitter functionality:

node main.js

Expected Outcome

When you run `main.js`, you should see the following output in the console:

```
User clicked a button!
Message received: Hello, world!
```

****