PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

GETTING STARTED WITH HTML

SOL. 1

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Sample HTML Program</title>
</head>
<body>
    <h1>HTML defines the content and structure of your website</h1>
</body>
</html>
```

OUTPUT:

HTML defines the content and structure of your website

SOL. 2

Comments in HTML are used to leave notes or explanations within the code that are not displayed in the browser. They help developers understand the structure of the HTML document, explain complex code, or temporarily disable code without deleting it. Comments are also useful for collaboration, making it easier for others (or your future self) to understand the code.

Purpose of Comments in HTML:

1. **Documentation: Explain what certain parts of the code do.**
2. **Organization: Break up sections of the code for better readability.**
3. **Debugging: Temporarily disable code to test other parts without removing it.**
4. **Collaboration: Provide context for other developers working on the same codebase.**

Syntax for Comments:

In HTML, comments are written using the following syntax:

<!-- This is a comment -->

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Comment Example</title>
</head>
<body>
    <!-- This is the main heading of the page -->
    <h1>Welcome to My Website</h1>

    <!-- This section is about my hobbies -->
    <h2>My Hobbies</h2>
    <p>I enjoy hiking, reading, and coding.</p>

    <!--
    The following section is for future projects.
    Uncomment the code below when ready to showcase projects.
    -->
    <!--
    <h2>My Projects</h2>
    <ul>
        <li>Project 1</li>
        <li>Project 2</li>
    </ul>
    -->

</body>
</html>
```

SOL. 3

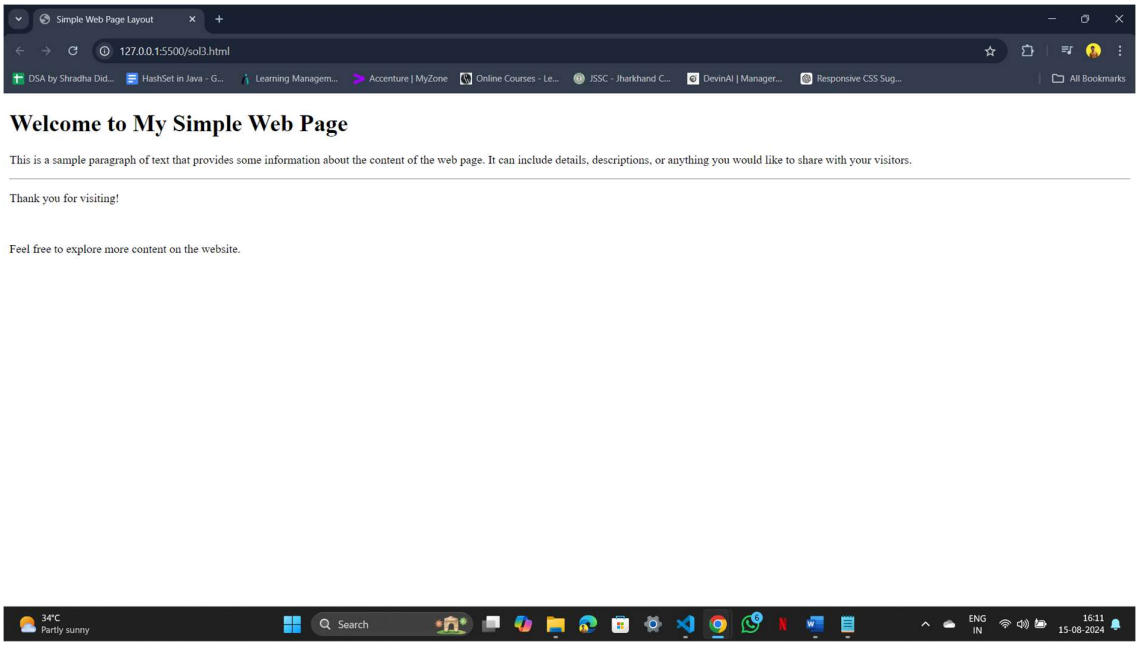
Index.html

```
<!DOCTYPE html>
<html lang="en">
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Simple Web Page Layout</title>
</head>

<body>
    <h1>Welcome to My Simple Web Page</h1>

    <p>
        This is a sample paragraph of text that provides some
information about the content of the web page.
        It can include details, descriptions, or anything you would
like to share with your visitors.
    </p>

    <hr> <!-- This creates a horizontal line -->

    <p>Thank you for visiting!</p>

    <br> <!-- This creates a line break -->

    <p>Feel free to explore more content on the website.</p>
</body>
</html>
```

Output:



SOL. 4

Tags and Elements in HTML

Tags and **elements** are fundamental concepts in HTML that help structure web content.

Tags:

A **tag** is a markup instruction that defines how content should be structured or displayed in an HTML document. Tags are enclosed in angle brackets (< >). Most tags come in pairs, consisting of an opening tag and a closing tag. The opening tag indicates the beginning of an element, while the closing tag (which includes a forward slash, e.g., </tagname>) indicates the end of that element.

Example of a Tag:

** This is a paragraph. **

In this example, is the opening tag, and is the closing tag.

Elements:

An **element** is a complete structure that consists of an opening tag, content, and a closing tag. Elements can contain text, other elements, or attributes that provide additional information about the element.

Example of an Element:

<h1>Welcome to My Website</h1>

In this example, the <h1> tag and the closing </h1> tag together form the **h1 element**. The content of the element is "Welcome to My Website."

Summary:

- **Tag**: The markup instructions (e.g., , <h1>) that define the content structure.
- **Element**: The combination of a start tag, content, and an end tag (e.g., <h1>Welcome to My Website</h1>).

Understanding tags and elements is essential for creating well-structured HTML documents.

SOL. 5

The DOCTYPE declaration in HTML is a special instruction that tells the web browser which version of HTML the document is using. It is placed at the very beginning of an HTML document and helps the browser render the content correctly. The DOCTYPE declaration is not an HTML tag; rather, it is an instruction that defines the document type.

Purpose of the DOCTYPE Declaration:

1. Rendering Mode: It informs the browser whether to render the page in "standards mode" or "quirks mode." Standards mode adheres to the rules of the specified HTML version, while quirks mode may allow for backward compatibility with older web pages.

2. Validation: It helps validators check the HTML code against the specified version to ensure it conforms to the rules and standards of that version.

Example of DOCTYPE Declaration:

For HTML5, the DOCTYPE declaration is very simple:

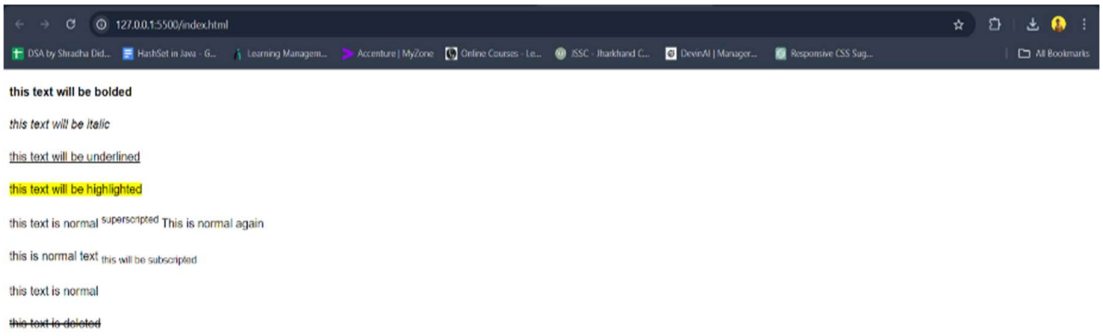
<!DOCTYPE html>

CORE HTML

SOL 1. –

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Formatted Text</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
        }

        .highlight {
            background-color: yellow;
        }
    </style>
</head>
<body>
    <p><strong>this text will be bolded</strong></p>
    <p><em>this text will be italic</em></p>
    <p><u>this text will be underlined</u></p>
    <p><span class="highlight">this text will be
highlighted</span></p>
    <p>this text is normal <sup>superscripted</sup> This is normal
again</p>
    <p>this is normal text <sub>this will be subscripted</sub></p>
    <p>this text is normal</p>
    <p><del>this text is deleted</del></p>
</body>
</html>
```

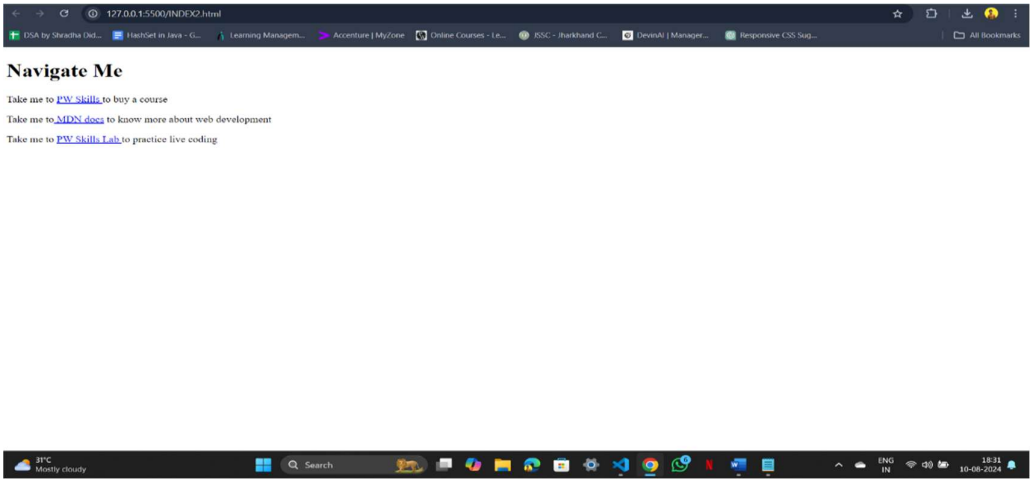
OUTPUT:



SOL 2. –

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Web Development Navigation</title>
</head>
<body>
    <h1>Navigate Me</h1>
    <p>Take me to <a href="https://www.pwskills.com/courses"
target="_blank">PW Skills </a>to buy a course</p>
    <p>Take me to<a href="https://developer.mozilla.org/"
target="_blank"> MDN docs</a> to know more about web
        development</p>
    <p>Take me to <a href="https://lab.pwskills.com/"
target="_blank">PW Skills Lab </a>to practice live coding</p>
</body>
</html>
```

OUTPUT:



SOL 3. –

index.html (Home Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Home - My Blog</title>
    <link rel="stylesheet" href="/index3/styles.css">
</head>
<body>
    <nav>
        <a href="index.html">Home</a>
        <a href="web-development.html">Web Development</a>
        <a href="web-design.html">Web Design</a>
    </nav>
    <div class="container">
        <h1>Welcome to My Blog</h1>
        <p>Hello! My name is Sourav Chakraborty, and this is my
personal blog where I share my journey and insights into
            the
            world of web development and design. Stay tuned for
updates and tips on creating beautiful, functional
            websites!</p>
    </div>
</body>
</html>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

web-development.html (Web Development Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Web Development - My Blog</title>
    <link rel="stylesheet" href="/index3/styles.css">
</head>
<body>
    <nav>
        <a href="index.html">Home</a>
        <a href="web-development.html">Web Development</a>
        <a href="web-design.html">Web Design</a>
    </nav>
    <div class="container">
        <h1>Web Development</h1>
        <p>Web development is the work involved in developing a
website for the Internet or an intranet. It can range
            from developing a simple single static page of plain text
to complex web applications, electronic
            businesses, and social network services. It involves both
frontend and backend development, and it's a
            constantly evolving field with new technologies emerging
all the time.</p>
    </div>
</body>
</html>
```

web-design.html (Web Design Page)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Web Design - My Blog</title>
    <link rel="stylesheet" href="/index3/styles.css">
</head>
<body>
    <nav>
        <a href="index.html">Home</a>
        <a href="web-development.html">Web Development</a>
        <a href="web-design.html">Web Design</a>
    </nav>
```

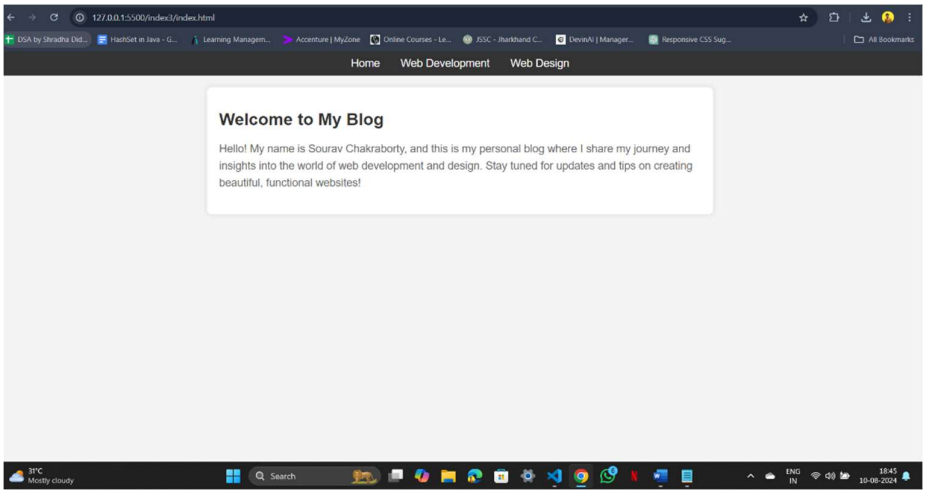
```
    <div class="container">
        <h1>Web Design</h1>
        <p>Web design refers to the design of websites that are
displayed on the internet. It usually refers to the user
            experience aspects of website development rather than
software development. Web design used to be focused on
            designing websites for desktop browsers; however, since
the mid-2010s, design for mobile and tablet browsers
            has become ever-increasingly important.</p>
    </div>
</body>
</html>
```

styles.css (CSS for all pages)

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
nav {
    background-color: #333;
    padding: 10px;
    text-align: center;
}
nav a {
    color: white;
    text-decoration: none;
    margin: 0 15px;
    font-size: 18px;
}
nav a:hover {
    text-decoration: underline;
}
.container {
    max-width: 800px;
    margin: 20px auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {
```

```
    color: #333;
    font-size: 28px;
    margin-bottom: 20px;
}
p {
    font-size: 18px;
    line-height: 1.6;
    color: #666;
}
```

OUTPUT:



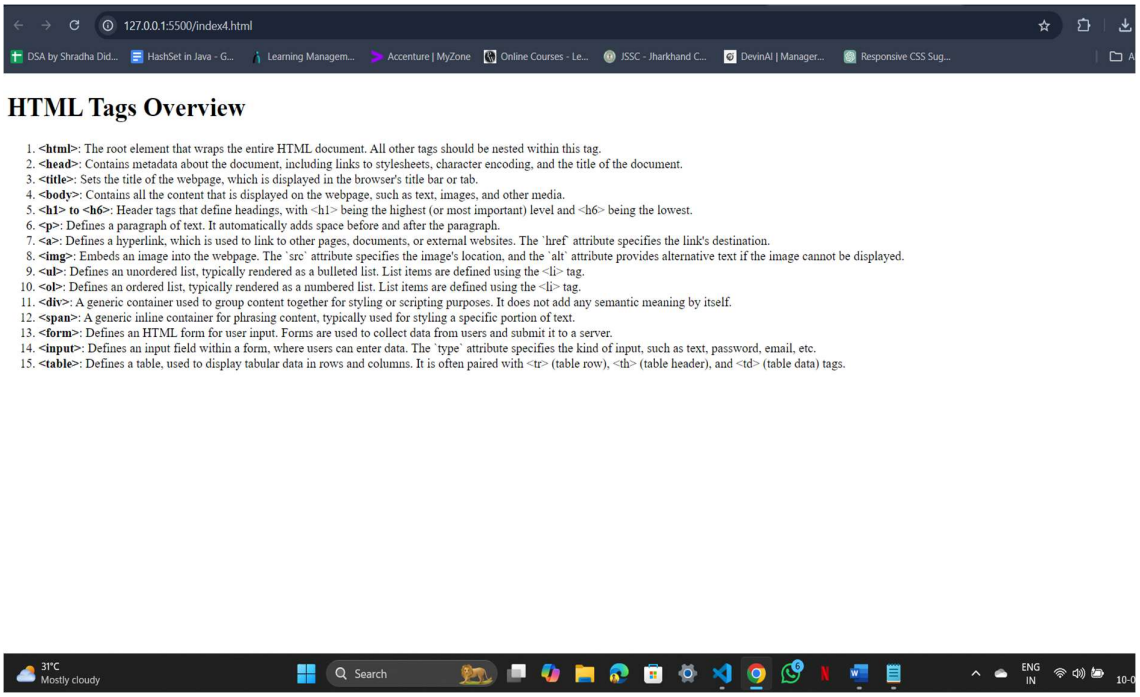
SOL 4. –

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>HTML Tags Overview</title>
```

```
</head>

<body>
    <h1>HTML Tags Overview</h1>
    <ol>
        <li>
            <strong>&lt;html&gt;</strong>: The root element that wraps
the entire HTML document. All other tags should
            be nested within this tag.
        </li>
        <li>
            <strong>&lt;head&gt;</strong>: Contains metadata about the
document, including links to stylesheets,
            character encoding, and the title of the document.
        </li>
        <li>
            <strong>&lt;title&gt;</strong>: Sets the title of the
webpage, which is displayed in the browser's title bar
            or tab.
        </li>
        <li>
            <strong>&lt;body&gt;</strong>: Contains all the content
that is displayed on the webpage, such as text,
            images, and other media.
        </li>
        <li>
            <strong>&lt;h1&gt; to &lt;h6&gt;</strong>: Header tags
that define headings, with &lt;h1&gt; being the
            highest (or most important) level and &lt;h6&gt; being the
lowest.
        </li>
        <li>
            <strong>&lt;p&gt;</strong>: Defines a paragraph of text.
It automatically adds space before and after the
            paragraph.
        </li>
        <li>
            <strong>&lt;a&gt;</strong>: Defines a hyperlink, which is
used to link to other pages, documents, or
            external websites. The `href` attribute specifies the
link's destination.
        </li>
        <li>
            <strong>&lt;img&gt;</strong>: Embeds an image into the
webpage. The `src` attribute specifies the image's
            location, and the `alt` attribute provides alternative
text if the image cannot be displayed.
        </li>
        <li>
            <strong>&lt;ul&gt;</strong>: Defines an unordered list,
typically rendered as a bulleted list. List items
```

```
            are defined using the &lt;li&gt; tag.
        </li>
        <li>
            <strong>&lt;ol&gt;</strong>: Defines an ordered list,
typically rendered as a numbered list. List items are
            defined using the &lt;li&gt; tag.
        </li>
        <li>
            <strong>&lt;div&gt;</strong>: A generic container used to
group content together for styling or scripting
            purposes. It does not add any semantic meaning by itself.
        </li>
        <li>
            <strong>&lt;span&gt;</strong>: A generic inline container
for phrasing content, typically used for styling a
            specific portion of text.
        </li>
        <li>
            <strong>&lt;form&gt;</strong>: Defines an HTML form for
user input. Forms are used to collect data from
            users and submit it to a server.
        </li>
        <li>
            <strong>&lt;input&gt;</strong>: Defines an input field
within a form, where users can enter data. The `type`
            attribute specifies the kind of input, such as text,
password, email, etc.
        </li>
        <li>
            <strong>&lt;table&gt;</strong>: Defines a table, used to
display tabular data in rows and columns. It is
            often paired with &lt;tr&gt; (table row), &lt;th&gt;
(table header), and &lt;td&gt; (table data) tags.
        </li>
    </ol>
</body>
</html>
```

OUTPUT

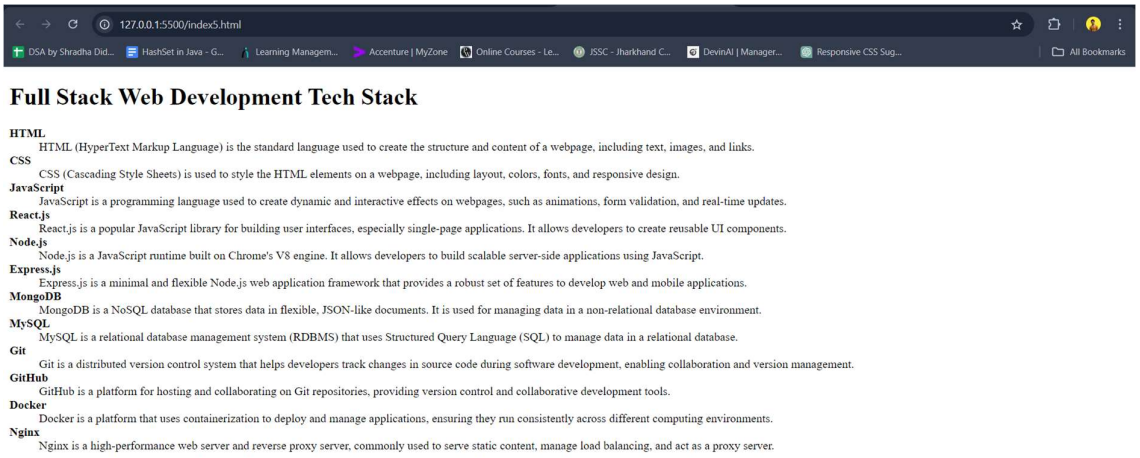


SOL 5. –

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Full Stack Web Development Tech Stack</title>
</head>
<body>
    <h1>Full Stack Web Development Tech Stack</h1>
    <dl>
        <dt><strong>HTML</strong></dt>
        <dd>HTML (HyperText Markup Language) is the standard language
used to create the structure and content of a
            webpage, including text, images, and links.</dd>

        <dt><strong>CSS</strong></dt>
        <dd>CSS (Cascading Style Sheets) is used to style the HTML
elements on a webpage, including layout, colors,
            fonts, and responsive design.</dd>

        <dt><strong>JavaScript</strong></dt>
        <dd>JavaScript is a programming language used to create
dynamic and interactive effects on webpages, such as
            animations, form validation, and real-time updates.</dd>
```

```
        <dt><strong>React.js</strong></dt>
        <dd>React.js is a popular JavaScript library for building user
interfaces, especially single-page applications.
            It allows developers to create reusable UI
components.</dd>

        <dt><strong>Node.js</strong></dt>
        <dd>Node.js is a JavaScript runtime built on Chrome's V8
engine. It allows developers to build scalable
            server-side applications using JavaScript.</dd>

        <dt><strong>Express.js</strong></dt>
        <dd>Express.js is a minimal and flexible Node.js web
application framework that provides a robust set of
            features to develop web and mobile applications.</dd>
        <dt><strong>MongoDB</strong></dt>
        <dd>MongoDB is a NoSQL database that stores data in flexible,
JSON-like documents. It is used for managing data
            in a non-relational database environment.</dd>

        <dt><strong>MySQL</strong></dt>
        <dd>MySQL is a relational database management system (RDBMS)
that uses Structured Query Language (SQL) to manage
            data in a relational database.</dd>

        <dt><strong>Git</strong></dt>
        <dd>Git is a distributed version control system that helps
developers track changes in source code during
            software development, enabling collaboration and version
management.</dd>

        <dt><strong>GitHub</strong></dt>
        <dd>GitHub is a platform for hosting and collaborating on Git
repositories, providing version control and
            collaborative development tools.</dd>

        <dt><strong>Docker</strong></dt>
        <dd>Docker is a platform that uses containerization to deploy
and manage applications, ensuring they run
            consistently across different computing environments.</dd>

        <dt><strong>Nginx</strong></dt>
        <dd>Nginx is a high-performance web server and reverse proxy
server, commonly used to serve static content,
            manage load balancing, and act as a proxy server.</dd>
    </dl>
</body>

</html>
```

OUTPUT:



SOL 6. –

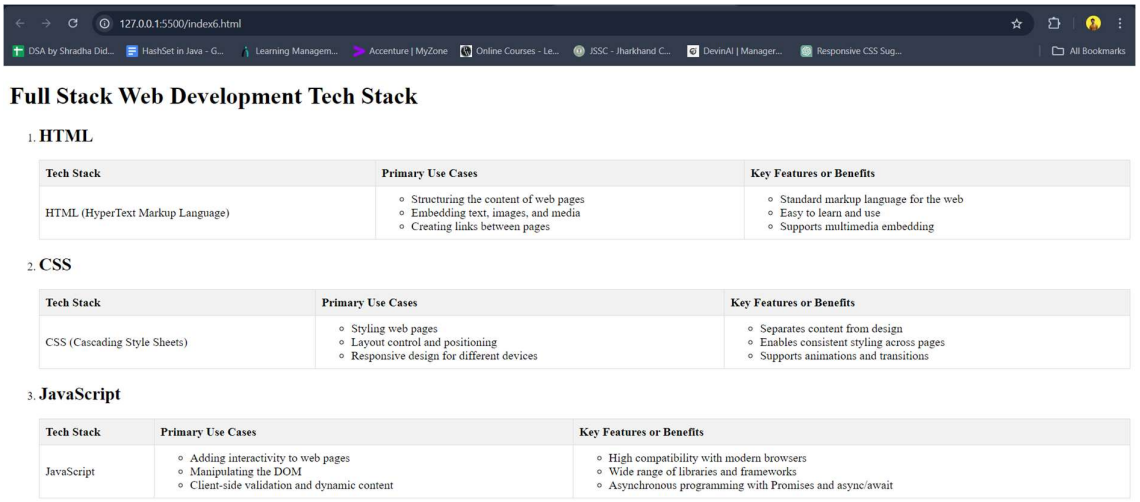
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Full Stack Web Development Tech Stack</title>
    <style>
        table {
            width: 100%;
            border-collapse: collapse;
            margin-bottom: 20px;
        }

        th,
        td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: left;
        }

        th {
            background-color: #f2f2f2;
        }
    </style>
</head>

<body>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

```
    <h1>Full Stack Web Development Tech Stack</h1>

    <ol>
        <li>
            <h2>HTML</h2>
            <table>
                <tr>
                    <th>Tech Stack</th>
                    <th>Primary Use Cases</th>
                    <th>Key Features or Benefits</th>
                </tr>
                <tr>
                    <td>HTML (HyperText Markup Language)</td>
                    <td>
                        <ul>
                            <li>Structuring the content of web
pages</li>
                            <li>Embedding text, images, and media</li>
                            <li>Creating links between pages</li>
                        </ul>
                    </td>
                    <td>
                        <ul>
                            <li>Standard markup language for the
web</li>
                            <li>Easy to learn and use</li>
                            <li>Supports multimedia embedding</li>
                        </ul>
                    </td>
                </tr>
            </table>
        </li>

        <li>
            <h2>CSS</h2>
            <table>
                <tr>
                    <th>Tech Stack</th>
                    <th>Primary Use Cases</th>
                    <th>Key Features or Benefits</th>
                </tr>
                <tr>
                    <td>CSS (Cascading Style Sheets)</td>
                    <td>
                        <ul>
                            <li>Styling web pages</li>
                            <li>Layout control and positioning</li>
                            <li>Responsive design for different
devices</li>
                        </ul>
                    </td>
```

```
                    <td>
                        <ul>
                            <li>Separates content from design</li>
                            <li>Enables consistent styling across
pages</li>
                            <li>Supports animations and
transitions</li>
                        </ul>
                    </td>
                </tr>
            </table>
        </li>

        <li>
            <h2>JavaScript</h2>
            <table>
                <tr>
                    <th>Tech Stack</th>
                    <th>Primary Use Cases</th>
                    <th>Key Features or Benefits</th>
                </tr>
                <tr>
                    <td>JavaScript</td>
                    <td>
                        <ul>
                            <li>Adding interactivity to web pages</li>
                            <li>Manipulating the DOM</li>
                            <li>Client-side validation and dynamic
content</li>
                        </ul>
                    </td>
                    <td>
                        <ul>
                            <li>High compatibility with modern
browsers</li>
                            <li>Wide range of libraries and
frameworks</li>
                            <li>Asynchronous programming with Promises
and async/await</li>
                        </ul>
                    </td>
                </tr>
            </table>
        </li>
    </ol>
</body>

</html>
```

OUTPUT:



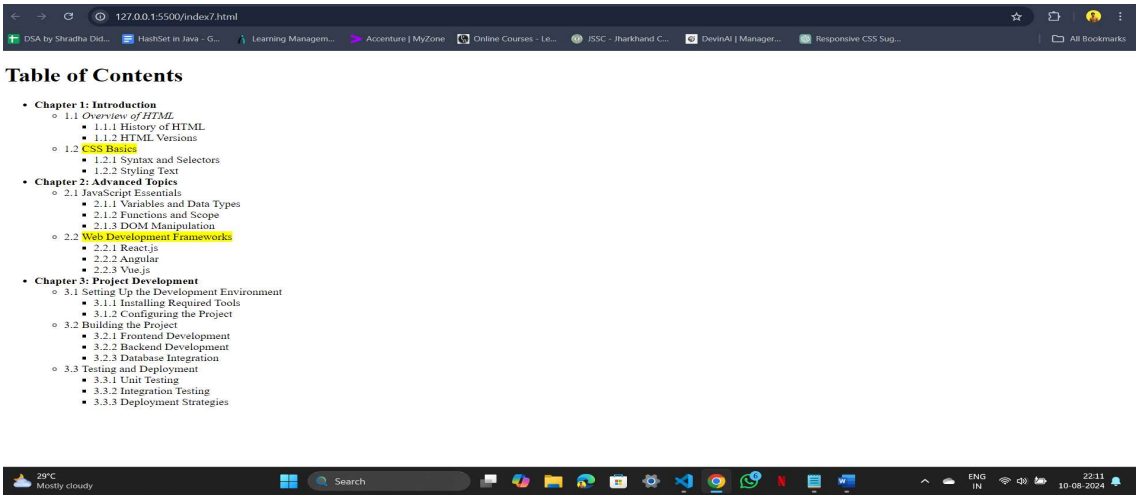
SOL. 7

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Table of Contents</title>
</head>
<body>
    <h1>Table of Contents</h1>
    <ul>
        <li><strong>Chapter 1: Introduction</strong></li>
        <ul>
            <li>1.1 <em>Overview of HTML</em></li>
            <ul>
                <li>1.1.1 History of HTML</li>
                <li>1.1.2 HTML Versions</li>
            </ul>
            <li>1.2 <mark>CSS Basics</mark></li>
            <ul>
                <li>1.2.1 Syntax and Selectors</li>
                <li>1.2.2 Styling Text</li>
            </ul>
        </ul>
        <li><strong>Chapter 2: Advanced Topics</strong></li>
        <ul>
            <li>2.1 JavaScript Essentials</li>
```

```
            <ul>
                <li>2.1.1 Variables and Data Types</li>
                <li>2.1.2 Functions and Scope</li>
                <li>2.1.3 DOM Manipulation</li>
            </ul>
            <li>2.2 <mark>Web Development Frameworks</mark></li>
            <ul>
                <li>2.2.1 React.js</li>
                <li>2.2.2 Angular</li>
                <li>2.2.3 Vue.js</li>
            </ul>
        </ul>
        <li><strong>Chapter 3: Project Development</strong></li>
        <ul>
            <li>3.1 Setting Up the Development Environment</li>
            <ul>
                <li>3.1.1 Installing Required Tools</li>
                <li>3.1.2 Configuring the Project</li>
            </ul>
            <li>3.2 Building the Project</li>
            <ul>
                <li>3.2.1 Frontend Development</li>
                <li>3.2.2 Backend Development</li>
                <li>3.2.3 Database Integration</li>
            </ul>
            <li>3.3 Testing and Deployment</li>
            <ul>
                <li>3.3.1 Unit Testing</li>
                <li>3.3.2 Integration Testing</li>
                <li>3.3.3 Deployment Strategies</li>
            </ul>
        </ul>
    </ul>
</body>
</html>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

OUTPUT:



SOL. 8

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Conference Schedule</title>
</head>
<body>
    <h1>Conference Schedule</h1>
    <table border="1" cellpadding="10">
        <tr>
            <th>Time</th>
            <th>Room 1</th>
            <th>Room 2</th>
            <th>Room 3</th>
        </tr>
        <tr>
            <td>09:00 - 10:00</td>
            <td colspan="3">Keynote Speech: The Future of Web
Development</td>
        </tr>
        <tr>
            <td>10:15 - 11:00</td>
            <td>Session A1: HTML5 Best Practices</td>
            <td>Session B1: Advanced CSS Techniques</td>
            <td rowspan="2">Workshop C1: Building Responsive
Layouts</td>
        </tr>
```

```
        <tr>
            <td>11:15 - 12:00</td>
            <td>Session A2: JavaScript Fundamentals</td>
            <td>Session B2: Introduction to Web Accessibility</td>
        </tr>
        <tr>
            <td>12:15 - 13:00</td>
            <td colspan="2">Panel Discussion: The State of Web
Design</td>
            <td>Session C2: UX/UI Design Principles</td>
        </tr>
        <tr>
            <td>13:00 - 14:00</td>
            <td colspan="3">Lunch Break</td>
        </tr>
        <tr>
            <td>14:15 - 15:00</td>
            <td>Session A3: Node.js for Beginners</td>
            <td rowspan="2">Workshop B3: Advanced JavaScript</td>
            <td>Session C3: Introduction to Web Security</td>
        </tr>
        <tr>
            <td>15:15 - 16:00</td>
            <td>Session A4: Deploying Web Applications</td>
            <td>Session C4: SEO Strategies for 2024</td>
        </tr>
        <tr>
            <td>16:15 - 17:00</td>
            <td colspan="3">Closing Remarks and Networking</td>
        </tr>
    </table>
</body>
</html>
```

OUTPUT:

Conference Schedule

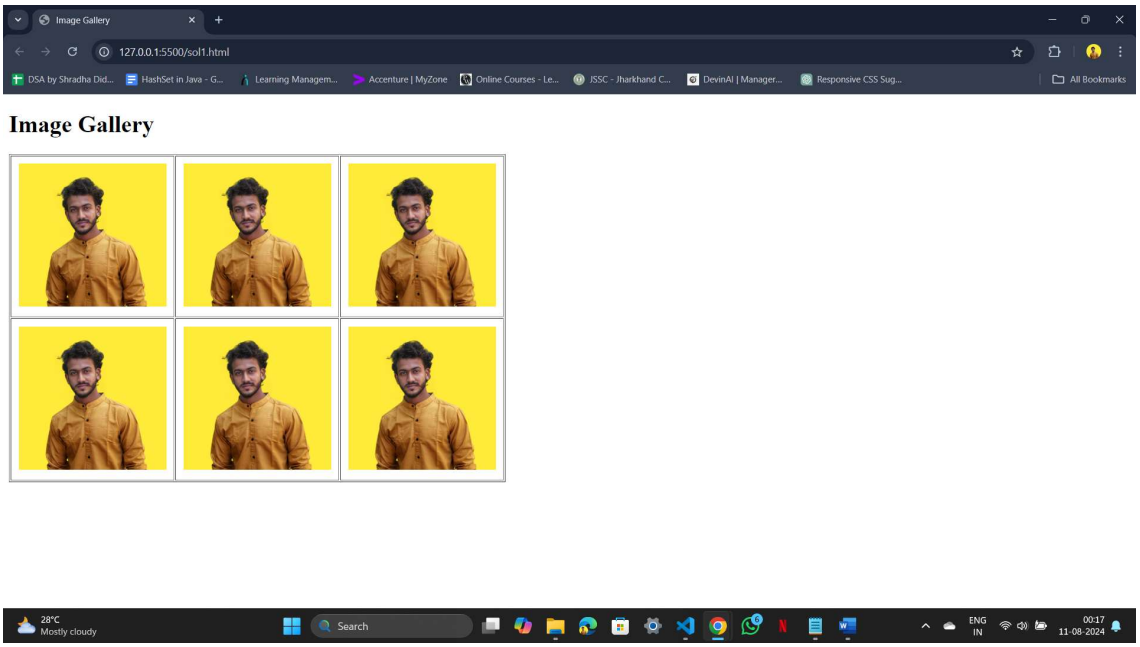
Time	Room 1	Room 2	Room 3
09:00 - 10:00	Keynote Speech: The Future of Web Development		
10:15 - 11:00	Session A1: HTML5 Best Practices	Session B1: Advanced CSS Techniques	Workshop C1: Building Responsive Layouts
11:15 - 12:00	Session A2: JavaScript Fundamentals	Session B2: Introduction to Web Accessibility	
12:15 - 13:00	Panel Discussion: The State of Web Design		Session C2: UX/UI Design Principles
13:00 - 14:00	Lunch Break		
14:15 - 15:00	Session A3: Node.js for Beginners	Workshop B3: Advanced JavaScript	Session C3: Introduction to Web Security
15:15 - 16:00	Session A4: Deploying Web Applications		Session C4: SEO Strategies for 2024
16:15 - 17:00	Closing Remarks and Networking		

Media and forms

SOL. 1

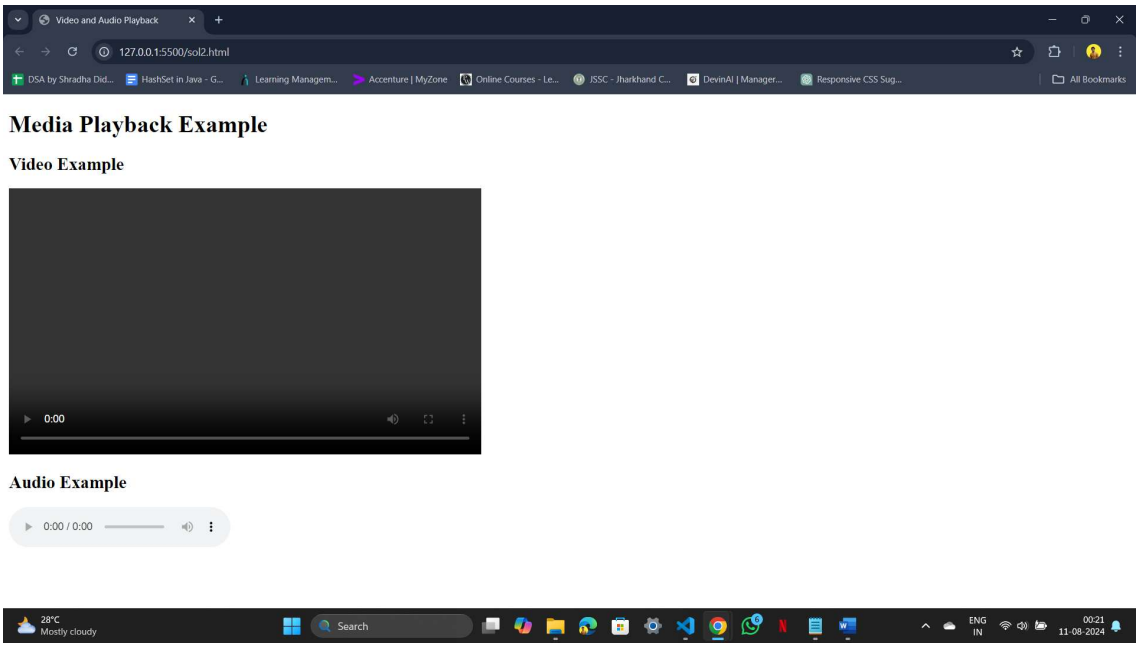
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Gallery</title>
</head>

<body>
  <h1>Image Gallery</h1>
  <table border="1" cellpadding="10">
    <tr>
      <td><img src="/sourav's pic.jpg" alt="Image 1" width="200"></td>
      <td><img src="/sourav's pic.jpg" alt="Image 2" width="200"></td>
      <td><img src="/sourav's pic.jpg" alt="Image 3" width="200"></td>
    </tr>
    <tr>
      <td><img src="/sourav's pic.jpg" alt="Image 4" width="200"></td>
      <td><img src="/sourav's pic.jpg" alt="Image 5" width="200"></td>
      <td><img src="/sourav's pic.jpg" alt="Image 6" width="200"></td>
    </tr>
  </table>
</body>
```

```
</html>
```



SOL. 2

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Video and Audio Playback</title>
</head>
<body>
    <h1>Media Playback Example</h1>
    <!-- Video Playback -->
    <h2>Video Example</h2>
    <video width="640" height="360" controls>
        <source src="example-video.mp4" type="video/mp4">
        <source src="example-video.ogg" type="video/ogg">
        Your browser does not support the video tag.
    </video>
    <!-- Audio Playback -->
    <h2>Audio Example</h2>
    <audio controls>
        <source src="example-audio.mp3" type="audio/mp3">
        <source src="example-audio.ogg" type="audio/ogg">
        Your browser does not support the audio element.
    </audio>
</body>
</html>
```



SOL. 3

To modify the previous example so that the video and audio play automatically when the page is loaded and loop infinitely, we can use the autoplay and loop attributes in the <video> and <audio> tags. Here's the updated code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Auto-Playing and Looping Media</title>
</head>
<body>
  <h1>Auto-Playing and Looping Media Example</h1>

  <!-- Video Playback -->
  <h2>Video Example</h2>
  <video width="640" height="360" controls autoplay loop>
    <source src="example-video.mp4" type="video/mp4">
    <source src="example-video.ogg" type="video/ogg">
    Your browser does not support the video tag.
  </video>
```

```
  <!-- Audio Playback -->
  <h2>Audio Example</h2>
  <audio controls autoplay loop>
    <source src="example-audio.mp3" type="audio/mp3">
    <source src="example-audio.ogg" type="audio/ogg">
    Your browser does not support the audio element.
  </audio>
</body>
</html>
```

SOL. 4

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Embed Physics Wallah Wikipedia Page</title>
</head>
<body>
  <h1>Physics Wallah Wikipedia Page</h1>
  <iframe src="https://en.wikipedia.org/wiki/Physics_Wallah" width="100%"
height="600"
    title="Physics Wallah Wikipedia Page">
    Your browser does not support iframes.
  </iframe>
</body>
</html>
```

Output:

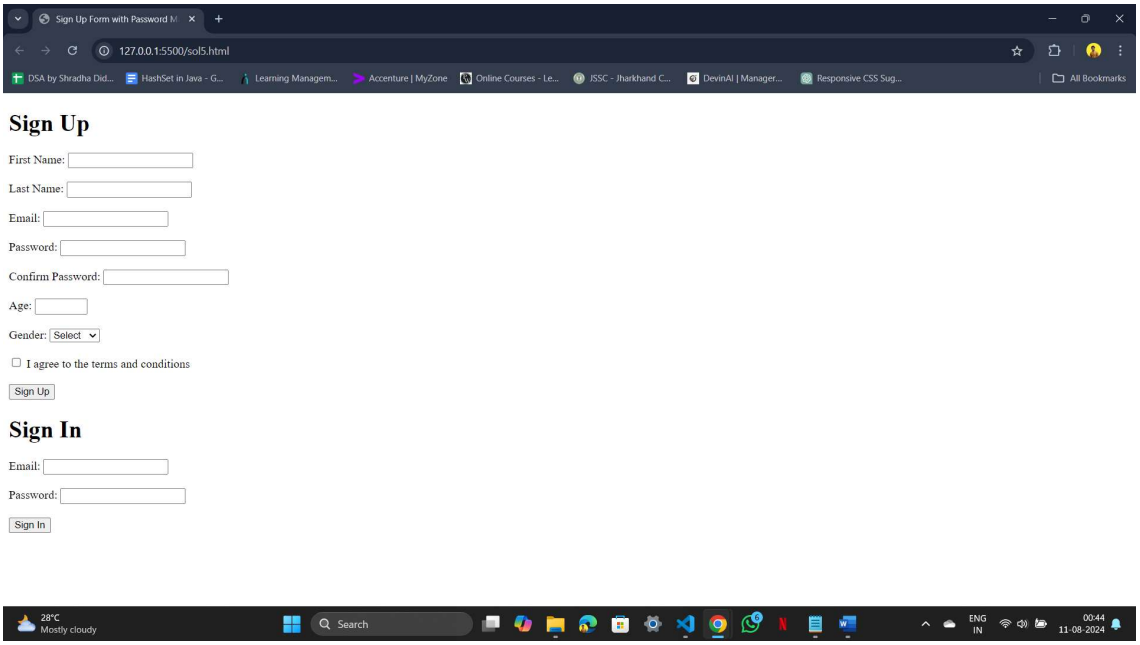
SOL. 5

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign Up Form with Password Matching</title>
</head>
<body>
    <h1>Sign Up</h1>
    <form
        oninput="confirmPassword.setCustomValidity(confirmPassword.value !=
password.value ? 'Passwords do not match.' : '')">
        <label for="firstName">First Name:</label>
        <input type="text" id="firstName" name="firstName" minlength="3"
required><br><br>
        <label for="lastName">Last Name:</label>
        <input type="text" id="lastName" name="lastName" minlength="3"
required><br><br>
        <label for="email">Email:</label>
```

```
    <input type="email" id="email" name="email" required><br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" minlength="6"
required><br><br>
    <label for="confirmPassword">Confirm Password:</label>
    <input type="password" id="confirmPassword" name="confirmPassword"
minlength="6" required><br><br>
    <label for="age">Age:</label>
    <input type="number" id="age" name="age" min="1" max="150"
required><br><br>

    <label for="gender">Gender:</label>
    <select id="gender" name="gender" required>
      <option value="">Select</option>
      <option value="Male">Male</option>
      <option value="Female">Female</option>
      <option value="Other">Other</option>
    </select><br><br>

    <input type="checkbox" id="terms" name="terms" required>
    <label for="terms">I agree to the terms and conditions</label><br><br>
    <input type="submit" value="Sign Up">
  </form>

  <h1>Sign In</h1>
  <form>
    <label for="signinEmail">Email:</label>
    <input type="email" id="signinEmail" name="signinEmail"
required><br><br>
    <label for="signinPassword">Password:</label>
    <input type="password" id="signinPassword" name="signinPassword"
minlength="6" required><br><br>
    <input type="submit" value="Sign In">
  </form>
</body>
</html>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

OUTPUT:



Starting with CSS

SOL 1. –

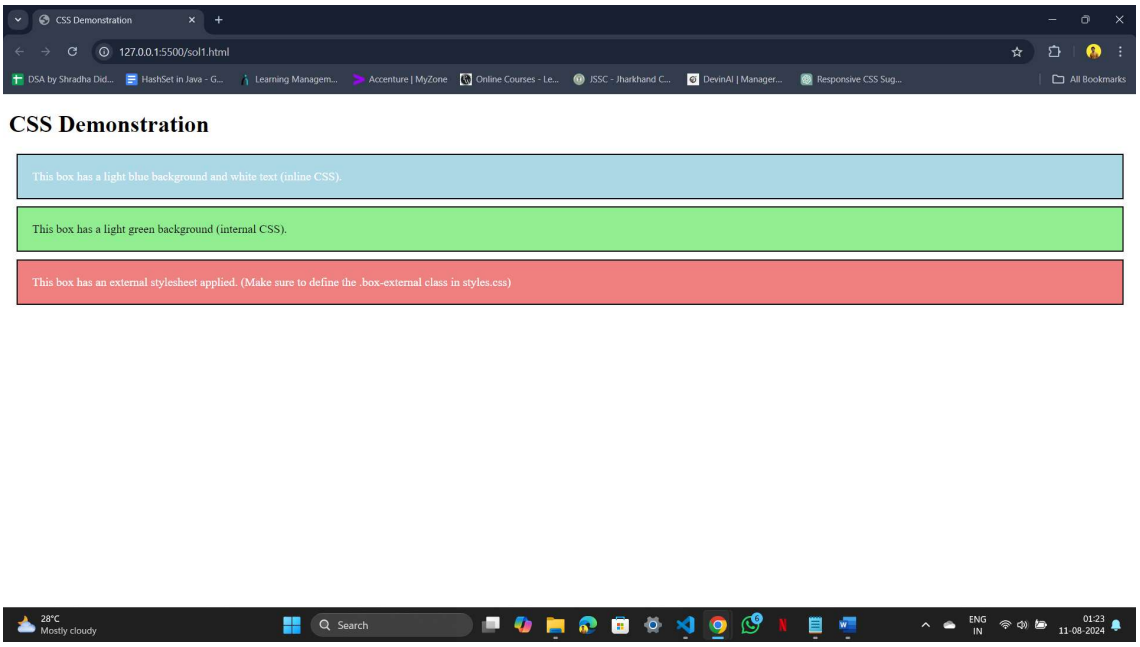
Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Demonstration</title>

    <!-- Internal CSS -->
    <style>
        /* This is an internal stylesheet */
        .box {
            padding: 20px;
            margin: 10px;
            border: 2px solid black;
        }

        /* Targeting a class with internal CSS */
        .box-blue {
```

```
        background-color: lightblue;
      }

      /* Targeting a class with internal CSS */
      .box-green {
        background-color: lightgreen;
      }
    </style>
    <!-- Link to external CSS file -->
    <link rel="stylesheet" href="styles.css"> <!-- External CSS file -->
</head>

<body>
    <h1>CSS Demonstration</h1>

    <!-- Div with inline CSS -->
    <div class="box box-blue" style="color: white;"> <!-- Inline CSS -->
        This box has a light blue background and white text (inline CSS).
    </div>

    <!-- Div with internal CSS -->
    <div class="box box-green">
        This box has a light green background (internal CSS).
    </div>

    <!-- Div with external CSS -->
    <div class="box box-external">
        This box has an external stylesheet applied. (Make sure to define the .box-external
class in styles.css)
    </div>
</body>
</html>
```

styles.css (External CSS)

```
/* External stylesheet */
.box-external {
    background-color: lightcoral;
    color: white; /* This will be overridden by inline styles if applied */
}
```



SOL. 2

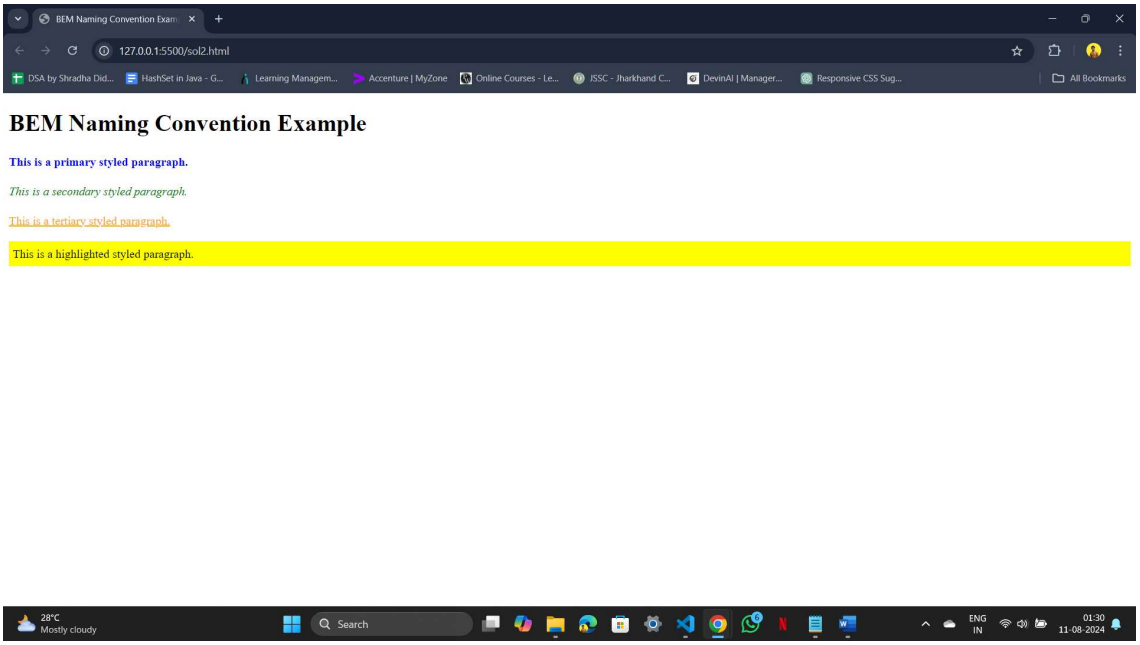
Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BEM Naming Convention Example</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <h1>BEM Naming Convention Example</h1>

    <p class="text text--primary">This is a primary styled paragraph.</p>
    <p class="text text--secondary">This is a secondary styled paragraph.</p>
    <p class="text text--tertiary">This is a tertiary styled paragraph.</p>
    <p class="text text--highlight">This is a highlighted styled paragraph.</p>
</body>
</html>
```

Style.css

```
/* Base block styling */
.text {
    font-size: 16px;
    line-height: 1.5;
```

```
}
/* Modifier styles */
.text--primary {
   color: blue;
   font-weight: bold;
}
.text--secondary {
   color: green;
   font-style: italic;
}
.text--tertiary {
   color: orange;
   text-decoration: underline;
}

.text--highlight {
   background-color: yellow;
padding: 5px;
}
```



SOL. 3

Index.html

```
<!DOCTYPE html>
<html lang="en">
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

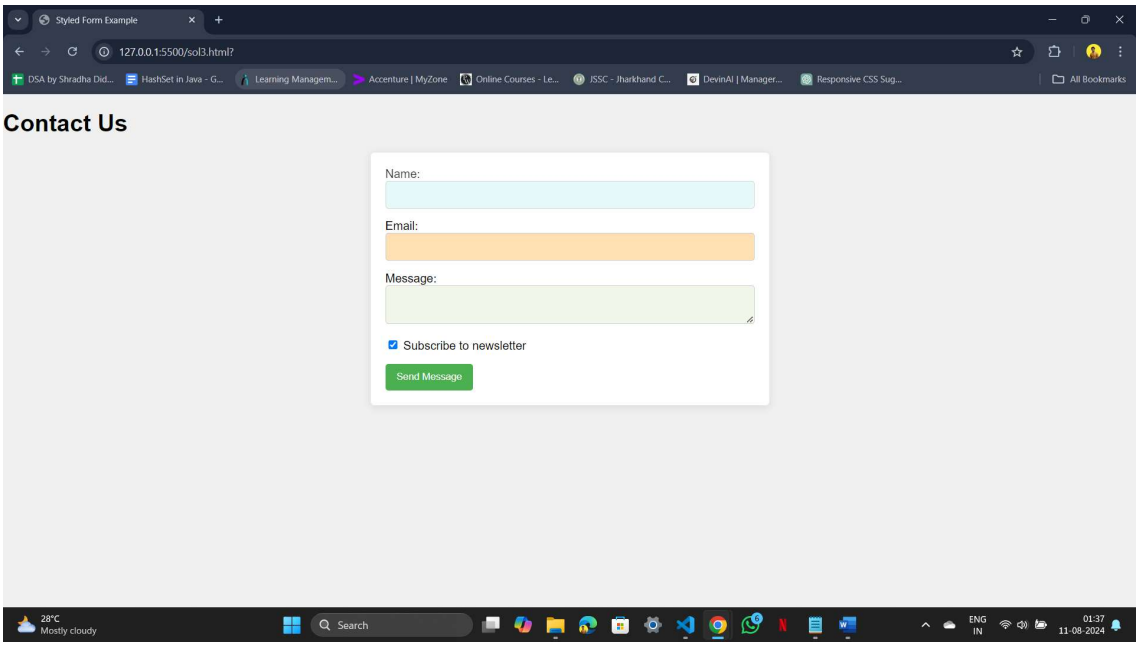
```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Styled Form Example</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <h1>Contact Us</h1>
  <form class="contact-form">
    <div class="form-section form-section--transparent">
      <label for="name">Name:</label>
      <input type="text" id="name" class="input input--primary" required>
    </div>

    <div class="form-section">
      <label for="email">Email:</label>
      <input type="email" id="email" class="input input--secondary" required>
    </div>

    <div class="form-section">
      <label for="message">Message:</label>
      <textarea id="message" class="input input--tertiary" required></textarea>
    </div>

    <div class="form-section">
      <label for="subscribe">
        <input type="checkbox" id="subscribe"> Subscribe to newsletter
      </label>
    </div>

    <button type="submit" class="button">Send Message</button>
  </form>
</body>
</html>
```

style.css

```
/* Basic reset */
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f0f0f0; /* Light background color for the page */
}
/* Styling the form */
.contact-form {
  max-width: 500px;
```

```
  margin: 20px auto;
  padding: 20px;
  background-color: #ffffff; /* White background for the form */
  border-radius: 5px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
/* Form section styling */
.form-section {
  margin-bottom: 15px;
}
/* Input field styling */
.input {
  width: 100%;
  padding: 10px;
  border-radius: 4px;
  border: 1px solid #ccc; /* Light border for inputs */
  box-sizing: border-box; /* Include padding in width */
}
/* Custom color palette */
.input--primary {
  background-color: #e0f7fa; /* Light cyan */
}
.input--secondary {
  background-color: #ffe0b2; /* Light orange */
}
.input--tertiary {
  background-color: #f1f8e9; /* Light green */
}
/* Button styling */
.button {
  background-color: #4caf50; /* Green button */
  color: white;
  padding: 10px 15px;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
/* Change button color on hover */
.button:hover {
  background-color: #45a049;
}
/* Apply opacity to one of the form sections */
.form-section--transparent {
  opacity: 0.8; /* Make this section slightly transparent */
}
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0



More on CSS

SOL. 1

Index.html

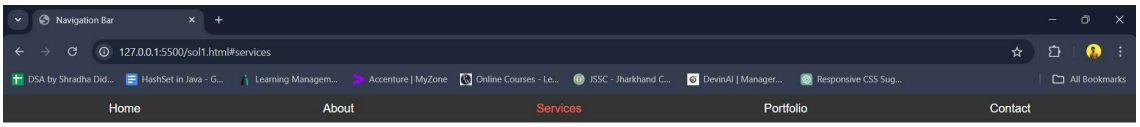
```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Navigation Bar</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <nav class="navbar">
        <ul class="nav-links">
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#services">Services</a></li>
            <li><a href="#portfolio">Portfolio</a></li>
            <li><a href="#contact">Contact</a></li>
        </ul>
    </nav>
</body>
```

```
</html>
```

Style.css

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
}

.navbar {
    background-color: #333; /* Dark background color */
    padding: 10px 20px; /* Padding around the navbar */
}

.nav-links {
    list-style-type: none; /* Remove bullet points */
    display: flex; /* Use flexbox for horizontal layout */
    justify-content: space-around; /* Evenly space links */
    margin: 0; /* Remove default margin */
    padding: 0; /* Remove default padding */
}

.nav-links li {
    margin: 0 15px; /* Spacing between items */
}

.nav-links a {
    color: white; /* Text color */
    text-decoration: none; /* Remove underline */
    padding: 10px 15px; /* Padding around the text */
    transition: color 0.3s; /* Smooth transition for color change */
}

.nav-links a:hover {
    color: #ff6347; /* Color on hover (tomato) */
}
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0



SOL. 2

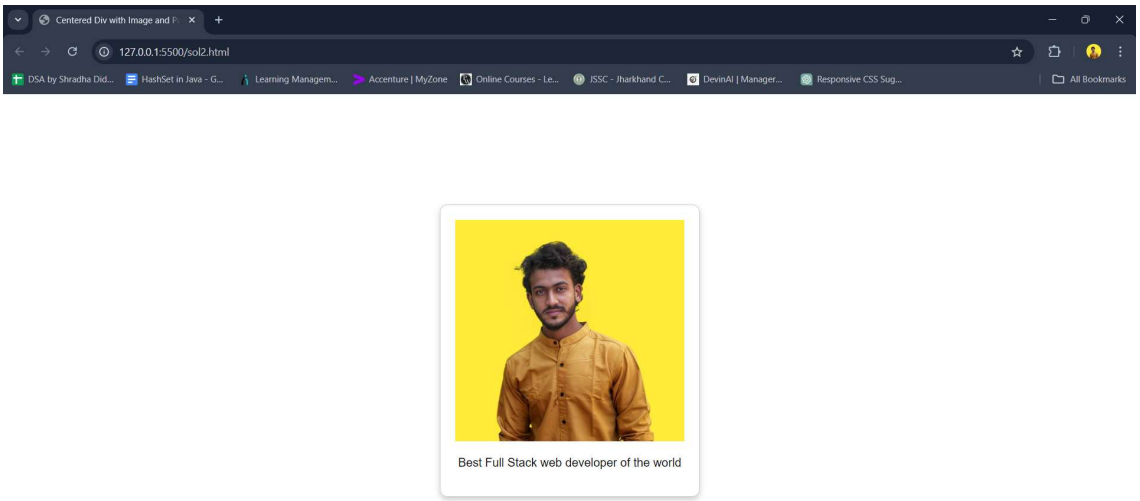
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Centered Div with Image and Paragraph</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="centered-container">
      <img src="image.jpg" alt="Description of Image" class="image">
      <p>  Best Full Stack web developer of the world</p>
   </div>
</body>
</html>
```

styles.css

```
body {
```

```
    margin: 0;
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center; /* Center horizontally */
    align-items: center; /* Center vertically */
    height: 100vh; /* Full viewport height */
}
.centered-container {
    text-align: center; /* Center text inside the div */
    padding: 20px; /* Padding around the content */
    border: 1px solid #ccc; /* Optional: border around the div */
    border-radius: 10px; /* Optional: rounded corners */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2); /* Optional: shadow effect */
}
.image {
    max-width: 100%; /* Responsive image */
    height: auto; /* Maintain aspect ratio */
}
```



SOL. 3

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Navigation Bar with Different Borders</title>
    <link rel="stylesheet" href="styles.css">
```

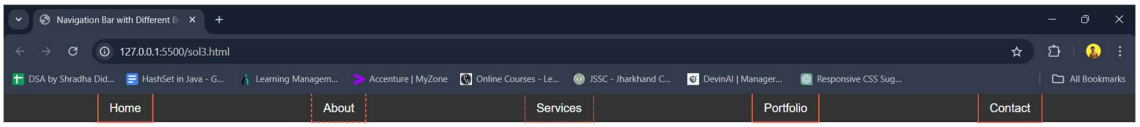
```
</head>
<body>
  <nav class="navbar">
    <ul class="nav-links">
      <li class="border-style1"><a href="#home">Home</a></li>
      <li class="border-style2"><a href="#about">About</a></li>
      <li class="border-style3"><a href="#services">Services</a></li>
      <li class="border-style4"><a href="#portfolio">Portfolio</a></li>
      <li class="border-style5"><a href="#contact">Contact</a></li>
    </ul>
  </nav>
</body>
</html>
```

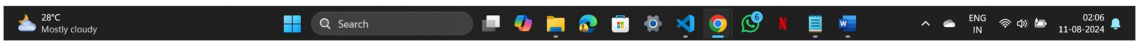
styles.css

```
body {
  margin: 0;
  font-family: Arial, sans-serif;
}

.navbar {
  background-color: #333; /* Dark background color */
  padding: 10px 20px; /* Padding around the navbar */
}

.nav-links {
  list-style-type: none; /* Remove bullet points */
  display: flex; /* Use flexbox for horizontal layout */
  justify-content: space-around; /* Evenly space links */
  margin: 0; /* Remove default margin */
  padding: 0; /* Remove default padding */
}

.nav-links li {
  margin: 0 15px; /* Spacing between items */
}

.nav-links a {
  color: white; /* Text color */
  text-decoration: none; /* Remove underline */
  padding: 10px 15px; /* Padding around the text */
  transition: color 0.3s; /* Smooth transition for color change */
  border: 2px solid transparent; /* Base border */
}

.nav-links a:hover {
  color: #ff6347; /* Color on hover (tomato) */
}
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

```
/* Different border styles for each navigation item */
.border-style1 a {
  border: 2px solid #ff6347; /* Solid border */
}

.border-style2 a {
  border: 2px dashed #ff6347; /* Dashed border */
}

.border-style3 a {
  border: 2px dotted #ff6347; /* Dotted border */
}

.border-style4 a {
  border: 2px double #ff6347; /* Double border */
}

.border-style5 a {
  border: 2px groove #ff6347; /* Groove border */
}

/* Additional hover effect for borders */
.nav-links li:hover a {
  border-color: white; /* Change border color on hover */
}
```





SOL. 4

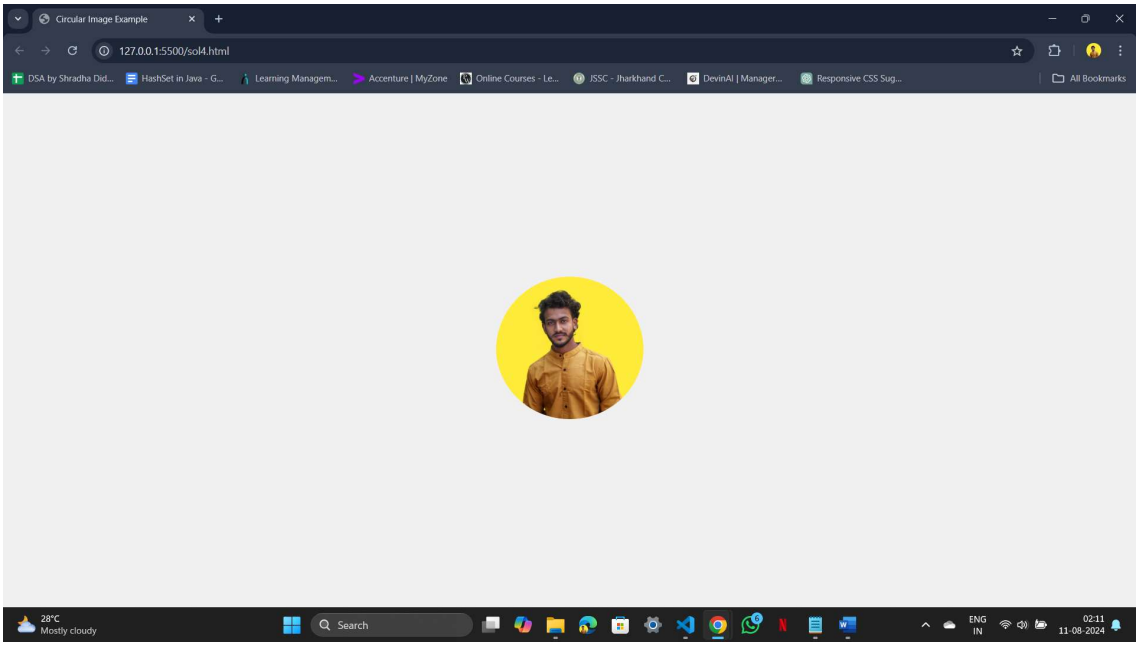
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Circular Image Example</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="image-container">
      <img src="image.jpg" alt="Description of Image" class="circular-image">
   </div>
</body>
</html>
```

styles.css

```
body {
   margin: 0;
   display: flex;
   justify-content: center; /* Center horizontally */
   align-items: center; /* Center vertically */
   height: 100vh; /* Full viewport height */
   background-color: #f0f0f0; /* Light background color */
}

.image-container {
   overflow: hidden; /* Hide overflow */
   width: 200px; /* Set width of the container */
   height: 200px; /* Set height of the container */
   border-radius: 50%; /* Make the container circular */
}

.circular-image {
   width: 100%; /* Make the image fill the container */
   height: auto; /* Maintain aspect ratio */
   border-radius: 50%; /* Make the image circular */
}
```

OUTPUT:



SOL. 5

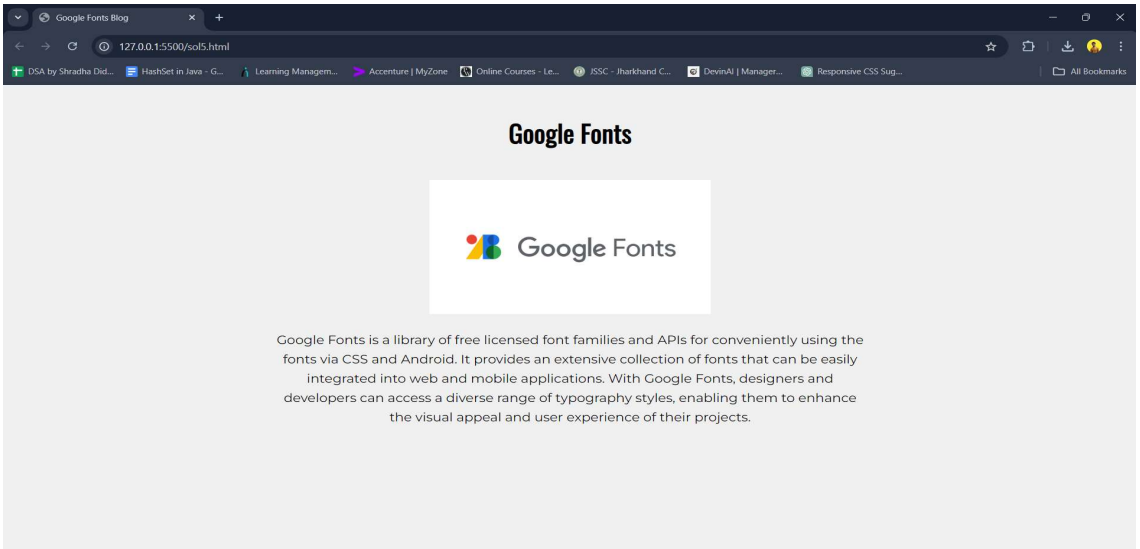
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Google Fonts Blog</title>
   <link rel="stylesheet" href="styles.css">
   <link
href="https://fonts.googleapis.com/css2?family=Oswald&family=Montserrat&display=swap"
rel="stylesheet">
</head>
<body>
   <div class="container">
      <h1 class="header">Google Fonts</h1>
      <img src="google-fonts-image.jpg" alt="Google Fonts" class="blog-image">
      <p class="description">Google Fonts is a library of free licensed font families and APIs for
conveniently using the fonts via CSS and Android. It provides an extensive collection of fonts that
can be easily integrated into web and mobile applications. With Google Fonts, designers and
developers can access a diverse range of typography styles, enabling them to enhance the visual
appeal and user experience of their projects.</p>
   </div>
</body>
</html>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

styles.css

```
body {
    margin: 0;
    font-family: Arial, sans-serif; /* Fallback font */
    background-color: #f0f0f0; /* Light background color */
}

.container {
    text-align: center; /* Center-align text */
    padding: 20px; /* Padding around the content */
}

.header {
    font-family: 'Oswald', sans-serif; /* Heading font */
    font-size: 36px; /* Font size for the heading */
}

.blog-image {
    max-width: 100%; /* Make image responsive */
    height: auto; /* Maintain aspect ratio */
    margin: 20px 0; /* Margin above and below the image */
}

.description {
    font-family: 'Montserrat', sans-serif; /* Paragraph font */
    font-size: 18px; /* Font size for the paragraph */
    line-height: 1.6; /* Line height for readability */
    max-width: 800px; /* Maximum width for paragraph */
    margin: 0 auto; /* Center the paragraph */
}
```



ASSIGNMENT MADE BY SOURAV CHAKRABORTY (souravchh456@gmail.com)

SOL. 6

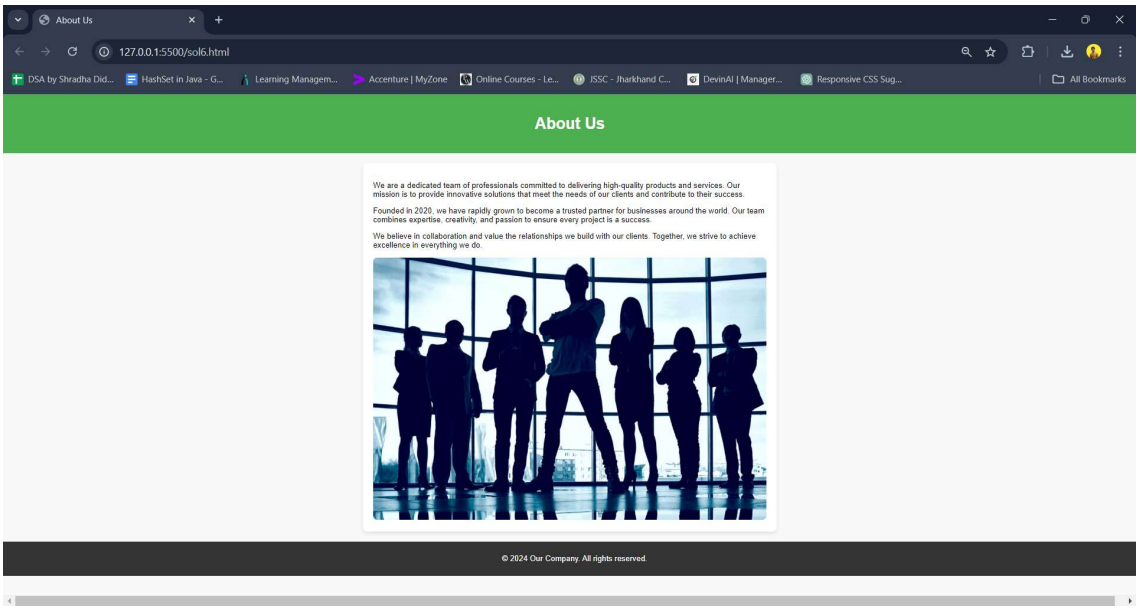
Index.html

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

    <p>We are a dedicated team of professionals committed to delivering high-quality products and services. Our
      mission is to provide innovative solutions that meet the needs of our clients and contribute to their
      success.</p>
    <p>Founded in 2020, we have rapidly grown to become a trusted partner for businesses around the world. Our team
      combines expertise, creativity, and passion to ensure every project is a success.</p>
    <p>We believe in collaboration and value the relationships we build with our clients. Together, we strive to
      achieve excellence in everything we do.</p>
    <img src="/team.png" alt="Our Team" class="team-image">
  </div>
  <footer>
    <p>&copy; 2024 Our Company. All rights reserved.</p>
  </footer>
</body>
</html>
```

styles.css

```
body {
  margin: 0;
  font-family: Arial, sans-serif;
  /* Fallback font */
  background-color: #f9f9f9;
  /* Light background color */
}
header {
  background-color: #4CAF50;
  /* Header background color */
```

```
    color: white;
    /* Header text color */
    text-align: center;
    /* Center-align header text */
    padding: 20px;
    /* Padding around the header */
}
.container {
    max-width: 800px;
    /* Maximum width for the content */
    margin: 20px auto;
    /* Center the container */
    padding: 20px;
    /* Padding around the content */
    background-color: white;
    /* White background for content */
    border-radius: 8px;
    /* Rounded corners */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    /* Shadow effect */
}
.team-image {
    max-width: 100%;
    /* Make image responsive */
    height: auto;
    /* Maintain aspect ratio */
    border-radius: 8px;
    /* Rounded corners for image */
}
footer {
    text-align: center;
    /* Center-align footer text */
    padding: 10px;
    /* Padding around the footer */
    background-color: #333;
    /* Footer background color */
    color: white;
    /* Footer text color */
    position: relative;
    /* Position relative for footer */
    bottom: 0;
    /* Stick to the bottom */
    width: 100%;
    /* Full width footer */
}
```



SOL. 7

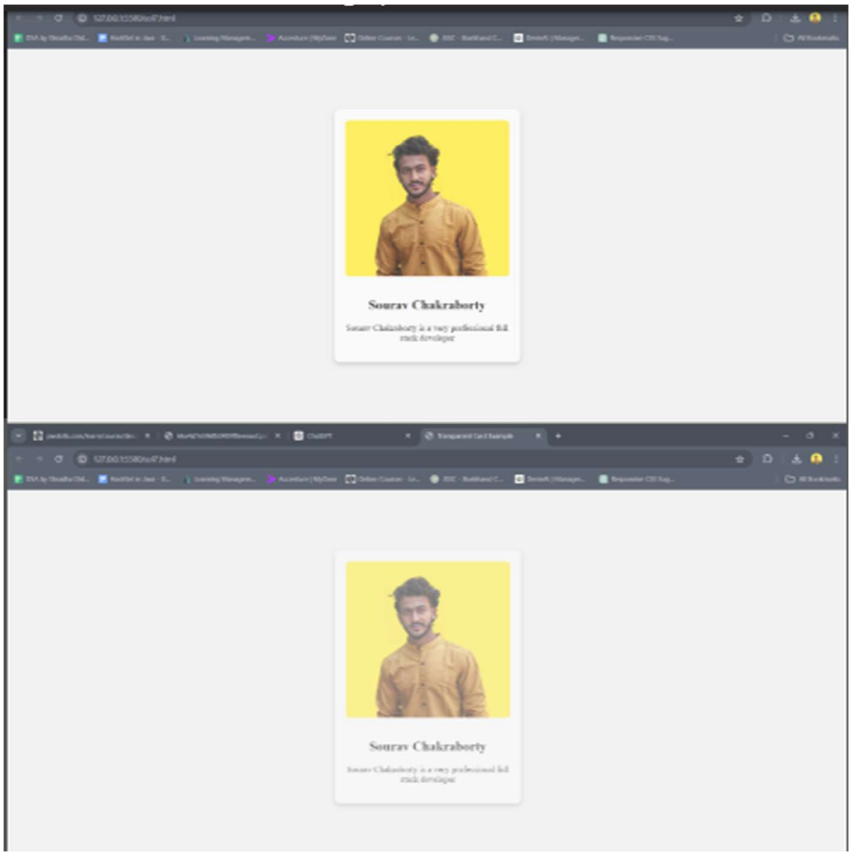
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Transparent Card Example</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="card">
      <img src="/sourav's pic.jpg" alt="Card Image" class="card-image"> <!-- Add your image here ->
      <h2>Sourav Chakraborty</h2>
      <p>Sourav Chakraborty is a  professional full stack developer</p>
   </div>
</body>
</html>
```

styles.css

```
body {
   margin: 0;
   display: flex;
   justify-content: center;
   /* Center horizontally */
```

```
    align-items: center;
    /* Center vertically */
    height: 100vh;
    /* Full viewport height */
    background-color: #f0f0f0;
    /* Light background color */
}
.card {
    background-color: rgba(255, 255, 255, 0.7);
    /* Semi-transparent white background */
    border-radius: 10px;
    /* Rounded corners */
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    /* Shadow effect */
    padding: 20px;
    /* Padding inside the card */
    text-align: center;
    /* Center-align text */
    width: 300px;
    /* Fixed width for the card */
    transition: opacity 0.3s ease;
    /* Smooth transition for opacity */
    opacity: 0.7;
    /* Initial opacity */
}
.card:hover {
    opacity: 1;
    /* Full opacity on hover */
}
.card-image {
    width: 100%;
    /* Make the image responsive */
    height: auto;
    /* Maintain aspect ratio */
    border-radius: 8px;
    /* Rounded corners for the image */
    margin-bottom: 15px;
    /* Margin below the image */
}
```



SOL. 8

Index.html

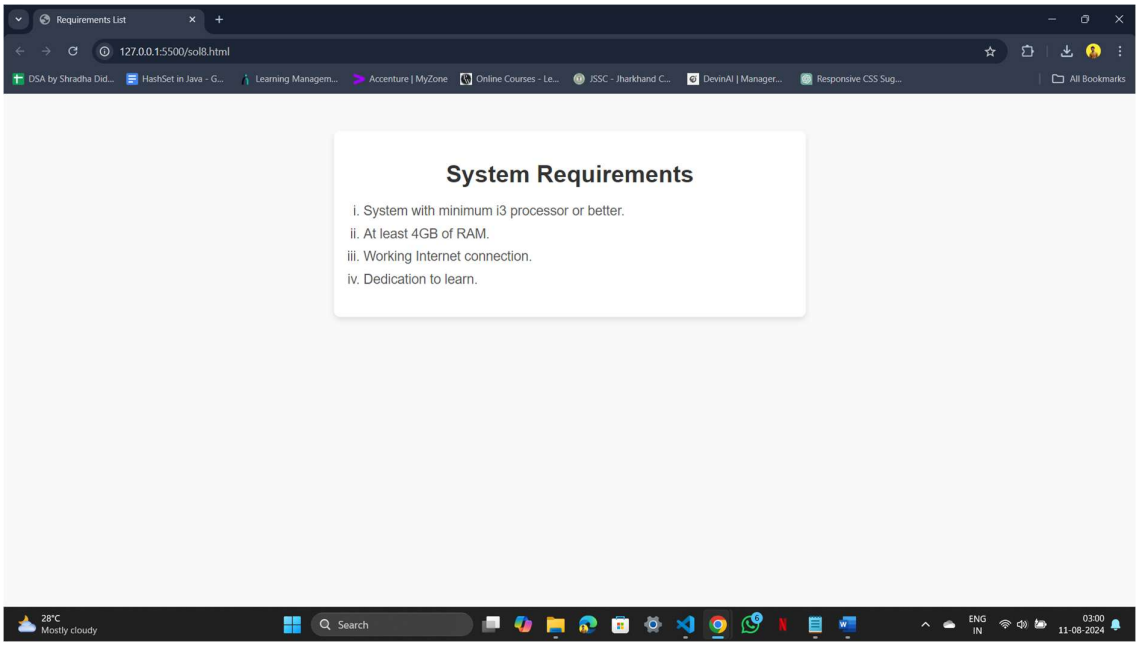
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Requirements List</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>System Requirements</h1>
    <ul class="requirements-list">
      <li>System with minimum i3 processor or better.</li>
      <li>At least 4GB of RAM.</li>
      <li>Working Internet connection.</li>
      <li>Dedication to learn.</li>
    </ul>
  </div>
</body>
```

```
</html>
```

styles.css

```
body {
   margin: 0;
   font-family: Arial, sans-serif; /* Fallback font */
   background-color: #f9f9f9; /* Light background color */
}

.container {
   max-width: 600px; /* Maximum width for the container */
   margin: 50px auto; /* Center the container */
   padding: 20px; /* Padding around the content */
   background-color: white; /* White background for content */
   border-radius: 8px; /* Rounded corners */
   box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); /* Shadow effect */
}

h1 {
   text-align: center; /* Center-align heading */
   color: #333; /* Dark heading color */
}

.requirements-list {
 list-style-type: lower-roman;
margin: 20px 0; /* Margin above and below the list */
   padding-left: 20px; /* Padding to indent the list */
}

.requirements-list li {
   font-size: 18px; /* Font size for list items */
   margin-bottom: 10px; /* Spacing between list items */
   color: #555; /* Slightly lighter text color */
}
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

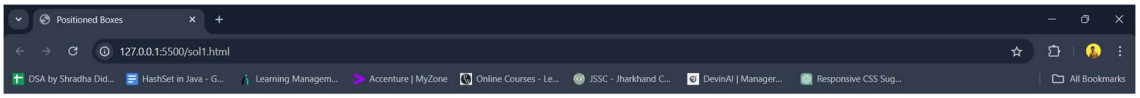


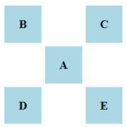
POSITIONS IN CSS

SOL. 1

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Positioned Boxes</title>
    <style>
        .box {
            width: 50px;
            height: 50px;
            position: absolute;
            background-color: lightblue;
            text-align: center;
            line-height: 50px;
            font-weight: bold;
        }

        /* Box A */
        #boxA {
            top: 200px;
            left: 200px;
        }

        /* Box B (-30px left, -30px above from the center of box A) */
```

```
    #boxB {
      top: calc(200px + 25px - 30px - 50px);
      left: calc(200px + 25px - 30px - 50px);
    }

    /* Box C (-30px right, -30px above from the center of box A) */
    #boxC {
      top: calc(200px + 25px - 30px - 50px);
      left: calc(200px + 25px + 30px);
    }

    /* Box D (-30px left, -30px below from the center of box A) */
    #boxD {
      top: calc(200px + 25px + 30px);
      left: calc(200px + 25px - 30px - 50px);
    }
    /* Box E (-30px right, -30px below from the center of box A) */
    #boxE {
      top: calc(200px + 25px + 30px);
      left: calc(200px + 25px + 30px);
    }
  </style>
</head>
<body>
  <div id="boxA" class="box">A</div>
  <div id="boxB" class="box">B</div>
  <div id="boxC" class="box">C</div>
  <div id="boxD" class="box">D</div>
  <div id="boxE" class="box">E</div>
</body>
</html>
```





SOL. 2

Absolute positioning and relative positioning are two key concepts in CSS that determine how elements are positioned on a webpage. Here's a breakdown of the differences between them:
Absolute Positioning

- **Position Relative to the Nearest Positioned Ancestor: An element with position: absolute is positioned relative to its nearest ancestor that has a position other than static (i.e., an ancestor with position: relative, absolute, or fixed). If no such ancestor exists, the element is positioned relative to the initial containing block (typically the browser window or the html element).**
- **Removes the Element from Normal Document Flow: When an element is positioned absolutely, it is removed from the normal document flow. This means it doesn't affect the layout of surrounding elements, and other elements will behave as if the absolutely positioned element doesn't exist.**
- **Precise Control Over Position: Absolute positioning allows you to specify the exact location of an element using the top, right, bottom, and left properties. This makes it possible to place the element precisely on the page.**
- **Example:**

```
<div style="position: relative;">
   <div style="position: absolute; top: 20px; left: 30px;">I'm absolutely positioned</div>
</div>
```

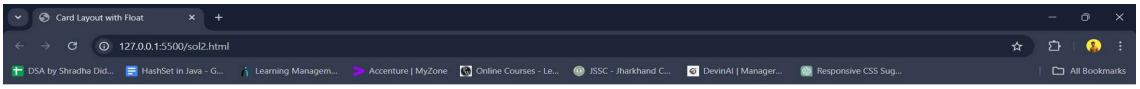
SOL. 3

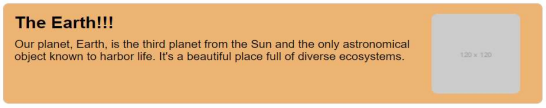
```
<!DOCTYPE html>
<html lang="en">
<head>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Card Layout with Float</title>
    <style>
        body {
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .card {
            width: 700px;
            height: auto;
            padding: 15px;
            border: 1px solid #ccc;
            border-radius: 8px;
            font-family: Arial, sans-serif;
            background-color: rgb(237, 180, 115);
        }
        .card img {
            width: 120px;
            height: auto;
            float: right;
            margin-right: 15px;
            border-radius: 8px;
        }
        .card h2 {
            margin-top: 0;
            margin-bottom: 10px;
        }
        .card p {
            margin: 0;
        }
        .clear {
            clear: both;
        }
    </style>
</head>
<body>
    <div class="card">
        <img src="https://via.placeholder.com/120" alt="Earth Image">
        <h2>The Earth!!!</h2>
        <p>Our planet, Earth, is the third planet from the Sun and the only astronomical object known
to harbor life.
            It's a beautiful place full of diverse ecosystems. </p>
        <div class="clear"></div>
    </div>
```

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



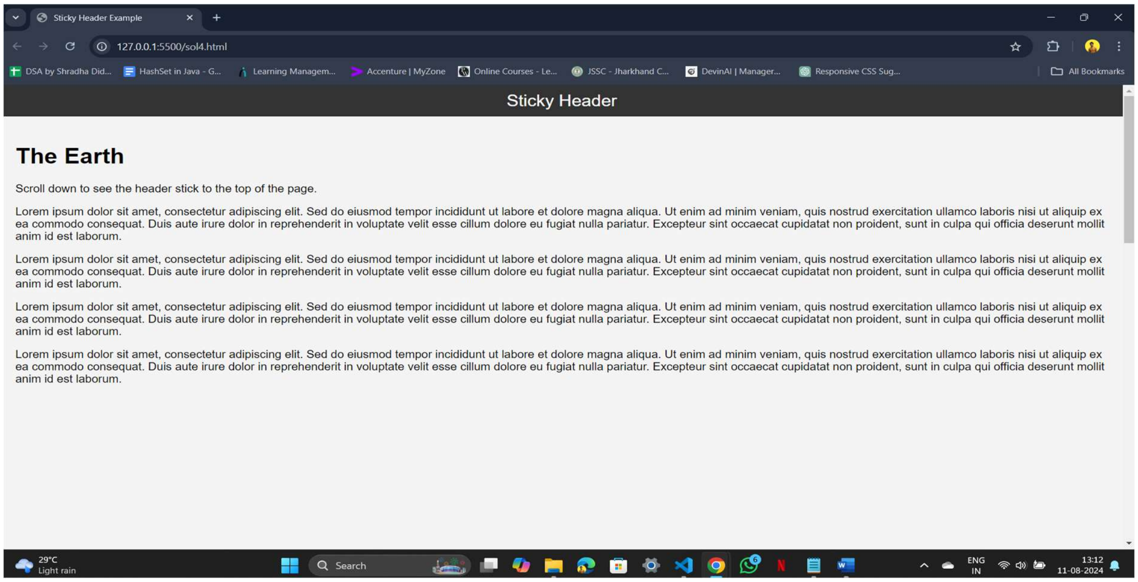


SOL. 4

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sticky Header Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    .header {
      background-color: #333;
      color: white;
      padding: 10px 0;
      text-align: center;
      font-size: 24px;
      position: sticky;
      top: 0;
      z-index: 1000;
    }
    .content {
      padding: 20px;
      height: 2000px;
      /* Added to demonstrate scrolling */
      background-color: #f4f4f4;
```

```
    }
  </style>
</head>
<body>
  <div class="header">
    Sticky Header
  </div>
  <div class="content">
    <h1>The Earth</h1>
    <p>Scroll down to see the header stick to the top of the page.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt
ut labore et dolore
      magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea
      commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat
      nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt mollit
      anim id est laborum.</p>
    <!-- More content to show scrolling -->
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt
ut labore et dolore
      magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea
      commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat
      nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt mollit
      anim id est laborum.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt
ut labore et dolore
      magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea
      commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat
      nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt molli  anim id est laborum.</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt
ut labore et dolore
      magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
aliquip ex ea
      commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum
dolore eu fugiat
      nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia
deserunt mollit
      anim id est laborum.</p>
    <!-- Repeat as needed -->
  </div>
```

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



SOL. 5

The z-index property in CSS controls the stacking order of elements that overlap each other. Elements with a higher z-index value will appear on top of elements with a lower z-index value. The z-index property only works on elements that have a position value other than static (e.g., relative, absolute, fixed, or sticky).

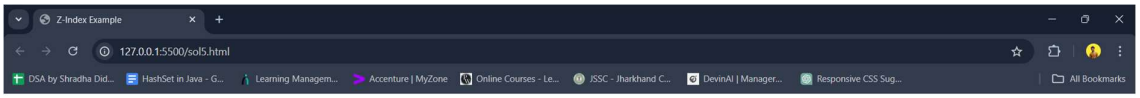
Key Points:

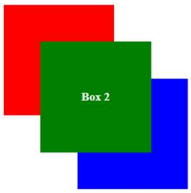
- **Higher z-index: Elements with higher z-index values are stacked on top of those with lower values.**
- **Requires Positioned Elements: For z-index to take effect, the element must be positioned (relative, absolute, fixed, or sticky).**

Example Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Z-Index Example</title>
  <style>
    .box {
      width: 150px;
      height: 150px;
      position: absolute;
      color: white;
      text-align: center;
      line-height: 150px;
      font-weight: bold;
    }
```

```
    #box1 {
      background-color: red;
      top: 50px;
      left: 50px;
      z-index: 1; /* Lower z-index */
    }

    #box2 {
      background-color: green;
      top: 100px;
      left: 100px;
      z-index: 3; /* Higher z-index */
    }

    #box3 {
      background-color: blue;
      top: 150px;
      left: 150px;
      z-index: 2; /* Middle z-index */
    }
  </style>
</head>
<body>
  <div id="box1" class="box">Box 1</div>
  <div id="box2" class="box">Box 2</div>
  <div id="box3" class="box">Box 3</div>
</body>
</html>
```





FLEXBOX IN CSS

SOL. 1

The main difference between the CSS Flexbox and CSS Grid layout models lies in how they handle layout design and their intended use cases.

CSS Flexbox

- **Purpose: Flexbox is primarily designed for one-dimensional layouts, meaning it works well when you need to arrange elements in a single row (horizontal) or column (vertical).**
- **Layout Direction: Flexbox excels when you need to control the layout along a single axis, either horizontal or vertical.**
- **Content-Based: The flex container adjusts the size of the items to fill the available space, which is helpful for dynamically resizing elements based on content.**
- **Use Case: Use Flexbox when you want to distribute space along a single line, align items along a single axis, or when you need to create responsive designs that adapt to different screen sizes.**

CSS Grid

- **Purpose: Grid is designed for two-dimensional layouts, allowing you to work with both rows and columns simultaneously.**
- **Layout Structure: CSS Grid allows you to define explicit grid lines and areas, making it easier to create complex layouts with both rows and columns.**
- **Layout-Based: Grid provides more control over the placement of items in a grid, enabling you to define specific rows, columns, and areas for your content.**
- **Use Case: Use Grid when you need to create more complex, structured layouts where both horizontal and vertical alignment of elements is important, or when you need to design entire page layouts.**

When to Use Each:

- **Use Flexbox: For simpler, one-dimensional layouts, such as navigation bars, aligning items in a single row or column, or distributing space between elements evenly.**

- **Use Grid: For more complex layouts that require both rows and columns, such as entire webpage layouts, dashboards, or any design where you need to control the placement of elements in both dimensions.**

In summary, Flexbox is ideal for simpler, one-directional layouts, while Grid is better suited for more complex, two-dimensional layouts.

SOL. 2

Here's an explanation of the role of each key property in the Flexbox layout model:

1. justify-content

- **Role: This property aligns flex items along the main axis (horizontal by default). It controls how extra space is distributed between and around flex items within a flex container.**
- **Values:**
  - **flex-start: Items are packed toward the start of the flex container.**
  - **flex-end: Items are packed toward the end of the flex container.**
  - **center: Items are centered in the flex container.**
  - **space-between: Items are evenly distributed, with the first item at the start and the last item at the end.**
  - **space-around: Items are evenly distributed with equal space around them.**

2. align-items

- **Role: This property aligns flex items along the cross axis (vertical by default). It determines how items are positioned relative to each other in the perpendicular direction to the main axis.**
- **Values:**
  - **flex-start: Items are aligned at the start of the cross axis.**
  - **flex-end: Items are aligned at the end of the cross axis.**
  - **center: Items are centered along the cross axis.**
  - **baseline: Items are aligned along their baseline.**
  - **stretch: Items are stretched to fill the container (this is the default).**

3. gap

- **Role: The gap property (previously known as grid-gap in Grid layout) defines the space between flex items. It adds a consistent spacing between items in both the main and cross axes.**
- **Values: It can take a single value (for both axes) or two values (first for the row gap, second for the column gap).**

4. flex-direction

- **Role: This property defines the direction in which flex items are placed in the flex container. It determines the main axis and can be set to display items in rows or columns.**
- **Values:**
  - **row: Items are placed in a row (default).**
  - **row-reverse: Items are placed in a row in reverse order.**
  - **column: Items are placed in a column.**
  - **column-reverse: Items are placed in a column in reverse order.**

5. flex-wrap

- **Role: This property determines whether flex items should wrap onto multiple lines or stay on a single line when they exceed the container's width or height.**
- **Values:**
  - **nowrap: All flex items are forced onto one line (default).**

- **wrap: Flex items will wrap onto multiple lines if necessary.**
- **wrap-reverse: Flex items will wrap onto multiple lines in reverse order.**

Summary

These properties work together to provide flexible and responsive layouts, allowing developers to control the alignment, spacing, and direction of flex items within a container effectively. By adjusting these properties, you can create a wide range of layouts that adapt to different screen sizes and content configurations.

SOL. 3

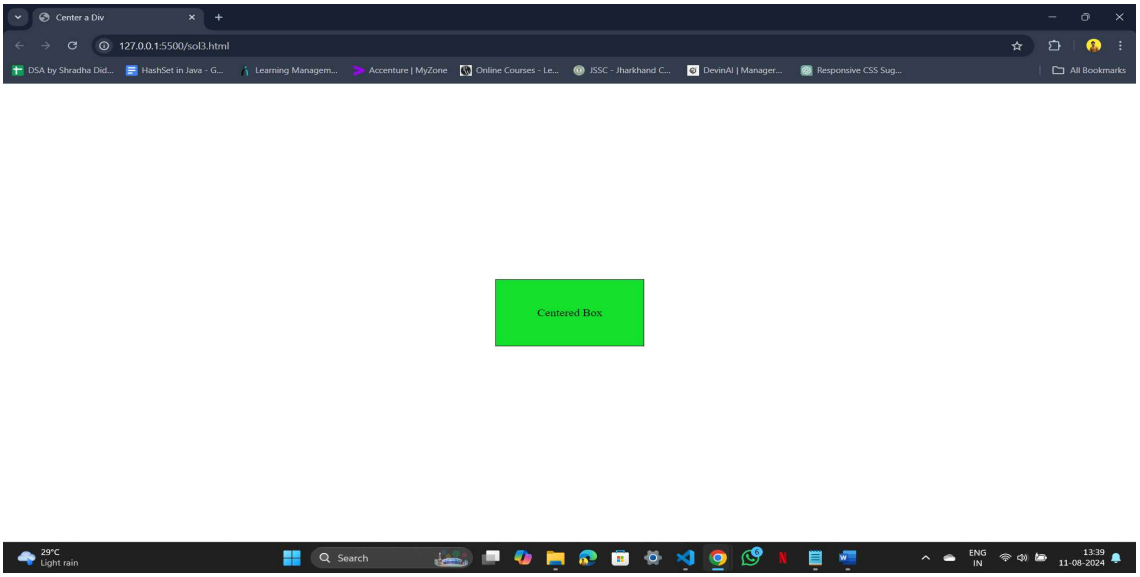
Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Center a Div</title>
   <link rel="stylesheet" href="styles.css">
</head>

<body>
   <div class="container">
      <div class="box">Centered Box</div>
   </div>
</body>

</html>
```

styles.css

```
html,
body {
   height: 100%;
   margin: 0;
}

.container {
   display: flex;
   /* Enable Flexbox */
   justify-content: center;
   /* Center horizontally */
   align-items: center;
   /* Center vertically */
   height: 100vh;
   /* Full viewport height */
}
```

```
.box {
   width: 200px;
   /* Width of the box */
   height: 100px;
   /* Height of the box */
   background-color: rgb(18, 224, 42);
   /* Background color */
   display: flex;
   /* Enable Flexbox for the box */
   justify-content: center;
   /* Center text horizontally */
   align-items: center;
   /* Center text vertically */
   border: 1px solid #000;
   /* Optional border */
}
```



SOL. 4

Pricing section based on a Figma design

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Pricing Section</title>
   <link rel="stylesheet" href="/sol4-style.css">
</head>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

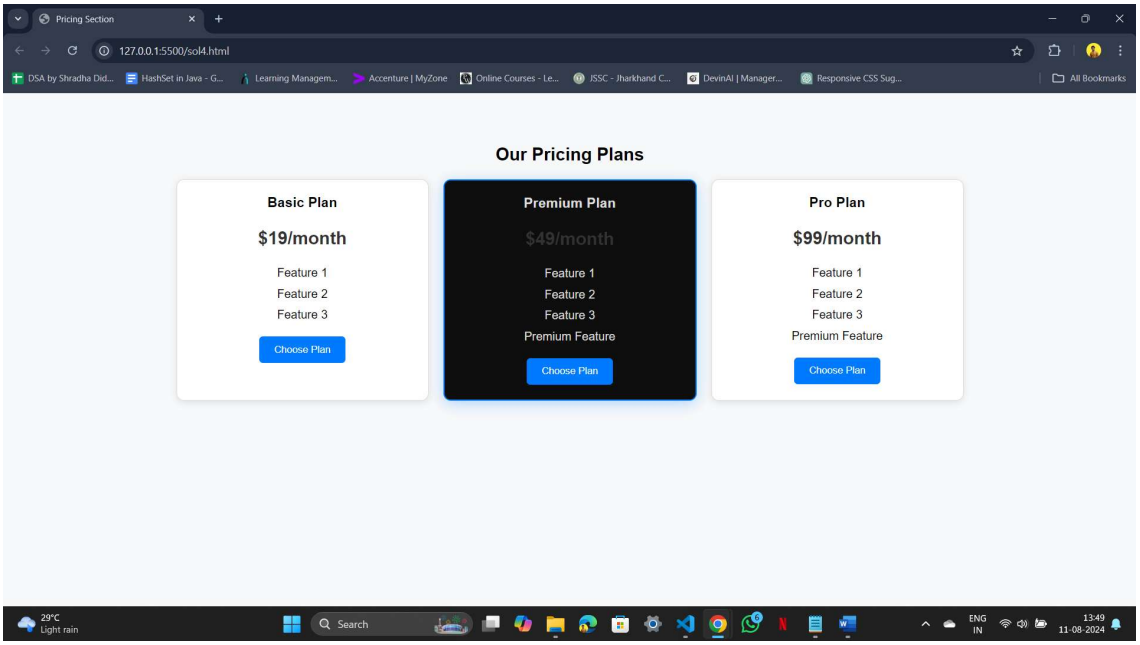
```
<body>
  <section class="pricing-section">
    <h2>Our Pricing Plans</h2>
    <div class="pricing-cards">
      <div class="pricing-card">
        <h3>Basic Plan</h3>
        <p class="price">$19/month</p>
        <ul>
          <li>Feature 1</li>
          <li>Feature 2</li>
          <li>Feature 3</li>
        </ul>
        <button>Choose Plan</button>
      </div>
      <div class="pricing-card premium">
        <h3>Premium Plan</h3>
        <p class="price">$49/month</p>
        <ul>
          <li>Feature 1</li>
          <li>Feature 2</li>
          <li>Feature 3</li>
          <li>Premium Feature</li>
        </ul>
        <button>Choose Plan</button>
      </div>
      <div class="pricing-card">
        <h3>Pro Plan</h3>
        <p class="price">$99/month</p>
        <ul>
          <li>Feature 1</li>
          <li>Feature 2</li>
          <li>Feature 3</li>
          <li>Premium Feature</li>
        </ul>
        <button>Choose Plan</button>
      </div>
    </div>
  </section>
</body>
</html>
```

styles.css

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 0;
  background-color: #f8f9fa;
}
```

```
.pricing-section {
    text-align: center;
    padding: 50px 20px;
}

.pricing-cards {
    display: flex;
    justify-content: center;
    gap: 20px;
    /* Space between cards */
}

.pricing-card {
    background-color: #fff;
    border: 1px solid #ddd;
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    width: 300px;
    /* Adjust as needed */
}

.pricing-card h3 {
    margin: 0 0 10px;
}

.price {
    font-size: 24px;
    font-weight: bold;
    color: #333;
}

.pricing-card ul {
    list-style: none;
    padding: 0;
    margin: 20px 0;
}

.pricing-card ul li {
    margin: 10px 0;
}

.pricing-card button {
    background-color: #007bff;
    /* Adjust based on design */
    color: #fff;
    border: none;
```

```
    border-radius: 5px;
    padding: 10px 20px;
    cursor: pointer;
}

.pricing-card button:hover {
    background-color: #0056b3;
    /* Darker shade for hover */
}
/* Highlight premium plan */
.premium {
    background-color: #0f0e0e;
    color: #ddd;
    border: 2px solid #007bff;
    /* Highlight the premium plan */
    box-shadow: 0 4px 20px rgba(0, 123, 255, 0.3);
}
```



SOL. 5

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>IRCTC Ticket Booking</title>
    <link rel="stylesheet" href="/sol5-style.css">
```

```
</head>

<body>
  <header>
    <img src="/irctc-logo.png" alt="IRCTC Logo" class="logo">
    <h1>IRCTC Train Ticket Booking</h1>
  </header>

  <main class="main-container">
    <form class="booking-form">
      <div class="form-group">
        <label for="from">From:</label>
        <input type="text" id="from" placeholder="Enter departure station" required>
      </div>
      <div class="form-group">
        <label for="to">To:</label>
        <input type="text" id="to" placeholder="Enter destination station" required>
      </div>
      <div class="form-group">
        <label for="date">Date of Journey:</label>
        <input type="date" id="date" required>
      </div>
      <div class="form-group">
        <label for="passengers">Number of Passengers:</label>
        <input type="number" id="passengers" min="1" value="1" required>
      </div>
      <button type="submit">Check Availability</button>
    </form>

    <img src="/rail.png" alt="Indian Railway Train" class="train-image">
  </main>

  <footer>
    <p>&copy; 2024 IRCTC. All rights reserved.</p>
  </footer>
</body>

</html>
```

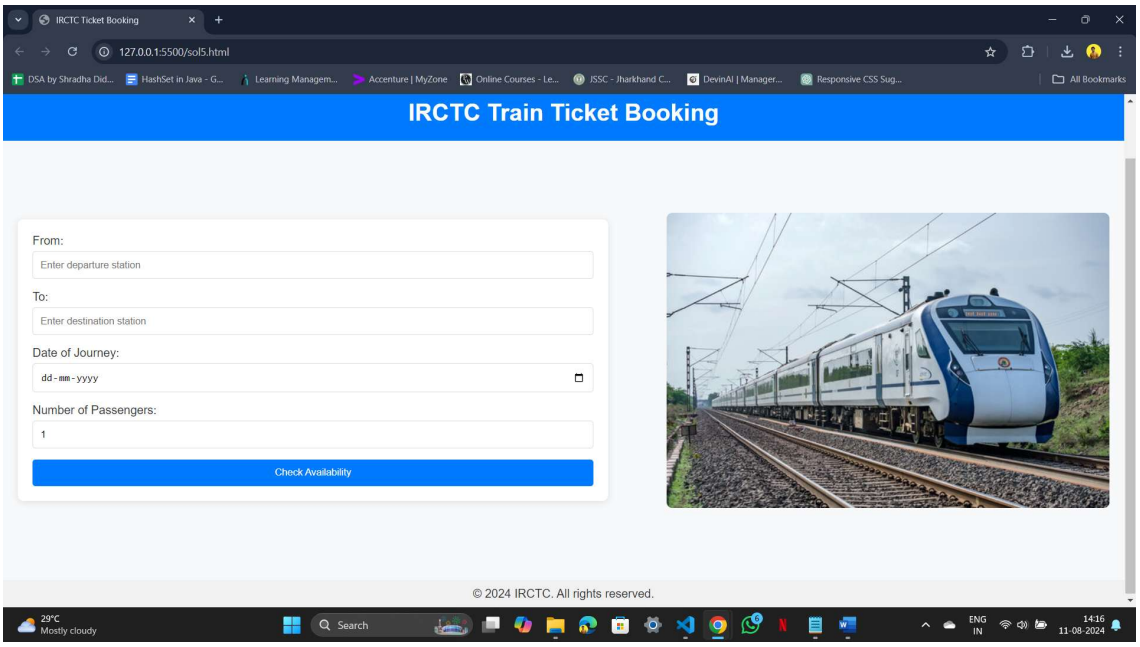
styles.css

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: Arial, sans-serif;
```

```
    background-color: #f8f9fa;
    color: #333;
}

header {
    background-color: #007bff;
    color: white;
    text-align: center;
    padding: 20px;
}

.logo {
    border-radius: 100%;
    width: 100px;
    /* Adjust size as needed */
    margin-bottom: 10px;
}

h1 {
    margin: 0;
}

.main-container {
    display: flex;
    /* Flexbox for layout */
    justify-content: space-between;
    /* Space between form and image */
    align-items: center;
    /* Center items vertically */
    padding: 20px;
    height: calc(100vh - 100px);
    /* Full height minus header and footer */
}

.booking-form {
    background-color: white;
    border-radius: 8px;
    padding: 20px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    width: 800px;
    /* Adjust width as needed */
}

.train-image {
    width: 600px;
    /* Set a fixed width for the image */
    max-width: 100%;
    /* Ensure responsiveness */
```

```
    border-radius: 8px;
    /* Rounded corners */
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input[type="text"],
input[type="date"],
input[type="number"] {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

footer {
    text-align: center;
    padding: 10px;
    background-color: #f1f1f1;
    position: relative;
    bottom: 0;
    width: 100%;
}
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0



GRID IN CSS

SOL. 1

Image gallery

Index.html

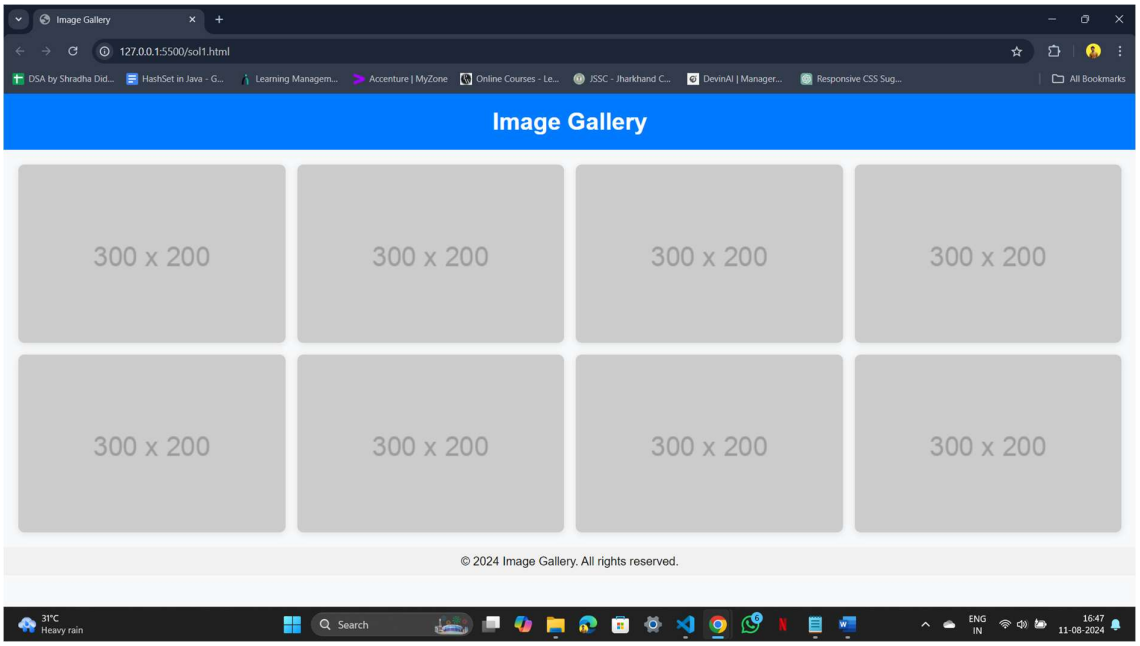
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Image Gallery</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <header>
    <h1>Image Gallery</h1>
  </header>
  <main class="gallery">
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 1">
    </div>
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 2">
    </div>
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 3">
    </div>
```

```
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 4">
    </div>
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 5">
    </div>
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 6">
    </div>
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 6">
    </div>
    <div class="gallery-item">
      <img src="https://via.placeholder.com/300x200" alt="Image 6">
    </div>
  </main>
  <footer>
    <p>&copy; 2024 Image Gallery. All rights reserved.</p>
  </footer>
</body>
</html>
```

styles.css

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
body {
  font-family: Arial, sans-serif;
  background-color: #f8f9fa;
}
header {
  background-color: #007bff;
  color: white;
  text-align: center;
  padding: 20px;
}
h1 {
  margin: 0;
}
.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  /* Responsive grid */
  gap: 16px;
  /* Space between grid items */
  padding: 20px;
}
```

```
.gallery-item {
    background-color: white;
    border-radius: 8px;
    overflow: hidden;
    /* Rounded corners */
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
.gallery-item img {
    width: 100%;
    /* Responsive image */
    height: auto;
    /* Maintain aspect ratio */
    display: block;
    /* Removes bottom space */
}
footer {
    text-align: center;
    padding: 10px;
    background-color: #f1f1f1;
}
```



SOL. 2

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
```

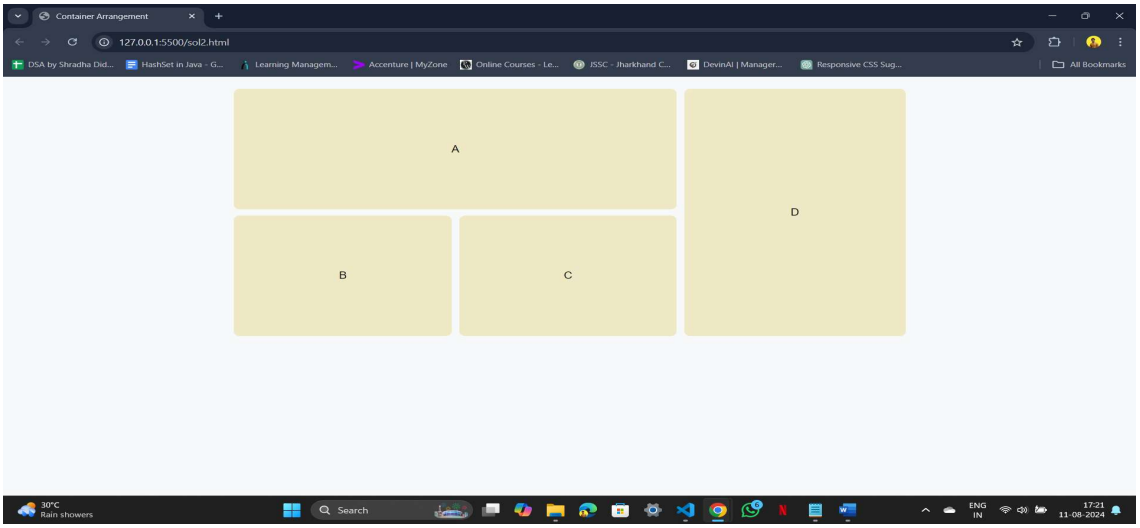
```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Container Arrangement</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body>
  <div class="container">
    <div class="box A">A</div>
    <div class="box B">B</div>
    <div class="box C">C</div>
    <div class="box D">D</div>
  </div>
</body>
</html>
```

styles.css

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: Arial, sans-serif;
  background-color: #f8f9fa;
}

.container {
  display: grid;
  grid-template-columns: 600px 300px;
  /* Box A is 600px, Box D is 300px wide */
  grid-template-rows: 200px 200px;
  /* Adjusted heights for Box A, B, C */
  gap: 10px;
  /* Space between boxes */
  width: 910px;
  /* Total width (600px + 300px + 10px gap) */
  margin: 20px auto;
  /* Center container */
}

.box {
  background-color: #f0e9c5;
  /* Blue background */
  color: rgb(0, 0, 0);
  display: flex;
  /* Use Flexbox for centering text */
```

```
  justify-content: center;
  /* Center horizontally */
  align-items: center;
  /* Center vertically */
  padding: 20px;
  border-radius: 8px;
}

.A {
  grid-column: 1 / 2;
  /* Occupy first column */
  grid-row: 1;
  /* Occupy first row */
  height: 200px;
  /* Set height */
  width: 600px;
  /* Set width */
}

.B {
  grid-column: 1 / 2;
  /* Occupy first column */
  grid-row: 2;
  /* Occupy second row */
  height: 200px;
  /* Set height */
  width: 295px;
  /* Set width */
}

.C {
  grid-column: 1 / 2;
  /* Occupy first column */
  grid-row: 2;
  /* Occupy second row */
  height: 200px;
  /* Set height */
  width: 295px;
  /* Set width */
  justify-self: end;
  /* Align to the right side of the column */
}
.D {
  grid-column: 2 / 3;
  /* Occupy second column */
  grid-row: 1 / 3;
  /* Span from first row to the third row */
  height: 410px;
```

```
  /* Combined height */
  width: 300px;
  /* Set width */
}
```



SOL. 3

The grid-auto-rows and grid-auto-columns properties in CSS Grid Layout are used to define the size of implicitly created grid rows and columns.

1. grid-auto-rows

This property specifies the size of rows that are automatically created when items are placed outside the explicitly defined grid.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>grid-auto-rows Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-columns: 100px 100px;
      grid-auto-rows: 50px;
      gap: 10px;
    }

    .item {
      background-color: lightblue;
      border: 1px solid #000;
    }

    .item:nth-child(1) { grid-row: 1 / 2; grid-column: 1 / 3; } /* First row spans 2 columns */
    .item:nth-child(2) { grid-row: 2; } /* Placed in the second row */
```

```
      .item:nth-child(3) { grid-row: 3; } /* Placed in the third row */
    </style>
</head>
<body>

<div class="grid-container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <div class="item">Item 4</div> <!-- This will go into an auto-created row -->
</div>

</body>
</html>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>grid-auto-rows Example</title>
    <style>
        .grid-container {
            display: grid;
            grid-template-columns: 100px 100px;
            grid-auto-rows: 50px;
            gap: 10px;
        }

        .item {
            background-color: lightblue;
            border: 1px solid #000;
        }

        .item:nth-child(1) { grid-row: 1 / 2; grid-column: 1 / 3; } /* First row spans 2 columns */
        .item:nth-child(2) { grid-row: 2; } /* Placed in the second row */
        .item:nth-child(3) { grid-row: 3; } /* Placed in the third row */
    </style>
</head>
<body>

<div class="grid-container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <div class="item">Item 4</div> <!-- This will go into an auto-created row -->
</div>
```

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

Explanation:

- **Grid Container: The grid has 2 columns of 100px each.**
- **grid-auto-rows: 50px; specifies that any automatically created row will be 50px tall.**
- **Items: Items 1, 2, and 3 are placed explicitly. Item 4 is placed in an automatically created row, which will also be 50px tall.**

2. grid-auto-columns

This property specifies the size of columns that are automatically created when items are placed outside the explicitly defined grid.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>grid-auto-columns Example</title>
  <style>
    .grid-container {
      display: grid;
      grid-template-rows: 100px 100px;
      grid-auto-columns: 50px;
      gap: 10px;
    }
    .item {
      background-color: lightgreen;
      border: 1px solid #000;
    }
    .item:nth-child(1) { grid-column: 1 / 2; grid-row: 1 / 3; } /* First column spans 2 rows */
    .item:nth-child(2) { grid-column: 2; } /* Placed in the second column */
    .item:nth-child(3) { grid-column: 3; } /* Placed in the third column */
  </style>
</head>
<body>
<div class="grid-container">
```

```
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <div class="item">Item 4</div> <!-- This will go into an auto-created column -->
</div>
</body>
</html>
```

SOL. 4

Index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Numbers</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
    <div class="item">5</div>
    <div class="item">6</div>
    <div class="item">7</div>
    <div class="item">8</div>
    <div class="item">9</div>

</body>
</html>
```

styles.css

```
body {
```

```
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
}

.item {
    width: 100px;
    height: 100px;
    background-color: #4CAF50;
    color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 24px;
    font-weight: bold;
    margin: 5px;
    border-radius: 10px;
}

.item:nth-child(even) {
    background-color: #2196F3;
}

.item:nth-child(odd) {
    background-color: #FF5722;
}
```

SOL. 5

Let's create a simple grid to demonstrate the difference between justify-items and justify-self.

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>ustify-items and justify-self</title>
   <link rel="stylesheet" href="styles.css">
</head>
<body>
   <div class="grid-container">
      <div class="item">1</div>
      <div class="item">2</div>
      <div class="item">3</div>
      <div class="item">4</div>
   </div>
</body>
</html>
```

styles.css

```
.grid-container {
   display: grid;
   grid-template-columns: repeat(2, 100px);
   grid-template-rows: repeat(2, 100px);
   gap: 10px;
   justify-items: center;
   /* Aligns all items horizontally to the center */
   background-color: #f0f0f0;
}
.item {
   background-color: #4CAF50;
   color: white;
   display: flex;
   justify-content: center;
   align-items: center;
   font-size: 24px;
   font-weight: bold;
}
/* Override justify-self for the second item */
.item:nth-child(2) {
   justify-self: end;
   /* Aligns this item to the right (end) */
}
/* Override justify-self for the third item */
.item:nth-child(3) {
   justify-self: start;
   /* Aligns this item to the left (start) */
}
```

Explanation:

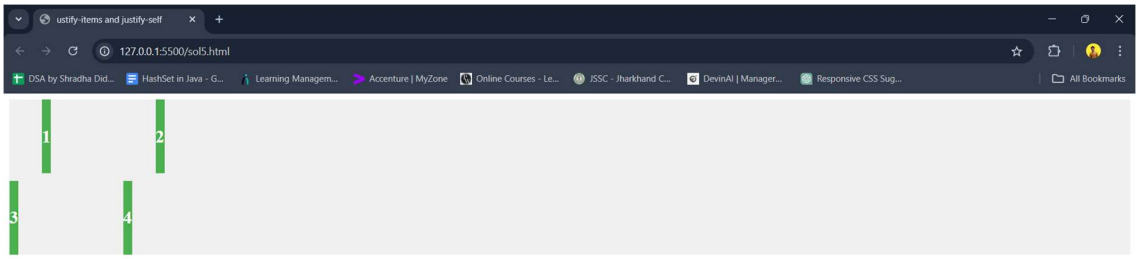
1. **Grid Container:**
   - **grid-template-columns: repeat(2, 100px);: Creates two columns, each 100px wide.**
   - **grid-template-rows: repeat(2, 100px);: Creates two rows, each 100px tall.**
   - **justify-items: center;: Aligns all items within their grid cells to the center horizontally.**
2. **Grid Items:**
   - **Item 1: Follows the justify-items: center rule and is centered in its grid cell.**
   - **Item 2: Uses justify-self: end;, which overrides the justify-items rule and aligns itself to the right (end) of its grid cell.**
   - **Item 3: Uses justify-self: start;, aligning itself to the left (start) of its grid cell.**
   - **Item 4: Like Item 1, it is centered due to the justify-items: center rule.**

Visual Layout:

- **Item 1 is centered.**
- **Item 2 is aligned to the right (end).**
- **Item 3 is aligned to the left (start).**
- **Item 4 is centered.**

Summary:

- **justify-items controls the horizontal alignment of all grid items within their cells.**
- **justify-self allows you to override the horizontal alignment for individual grid items, providing more granular control.**



Responsive Design

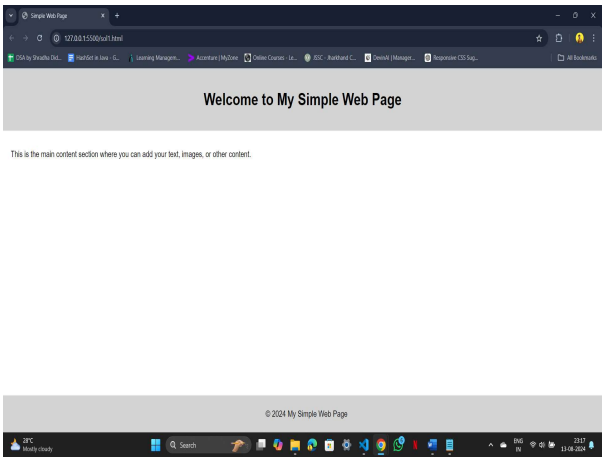
SOL. 1

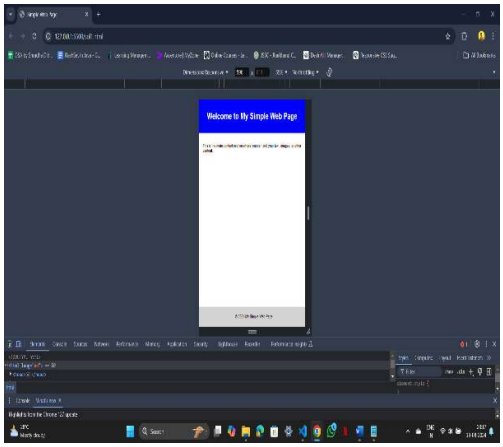
Index.html

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Web Page</title>
  <style>
    /* Basic styling */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    header {
      background-color: lightgray;
      text-align: center;
      padding: 20px;
    }
    main {
      padding: 20px;
    }
    footer {
      background-color: lightgray;
      text-align: center;
      padding: 10px;
      position: fixed;
      bottom: 0;
      width: 100%;
    }
    /* Media query for screens smaller than 600px */
    @media (max-width: 600px) {
      header {
        background-color: blue;
        color: white;
      }
    }
  </style>
</head>
<body>
  <header>
    <h1>Welcome to My Simple Web Page</h1>
  </header>
  <main>
    <p>This is the main content section where you can add your text, images, or other
content.</p>
  </main>
  <footer>
    <p>&copy; 2024 My Simple Web Page</p>
```

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



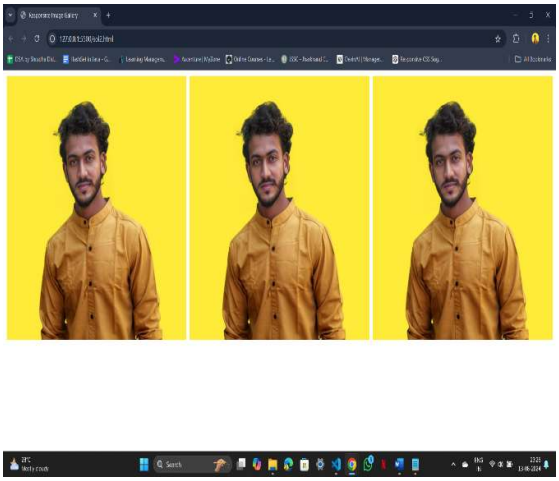


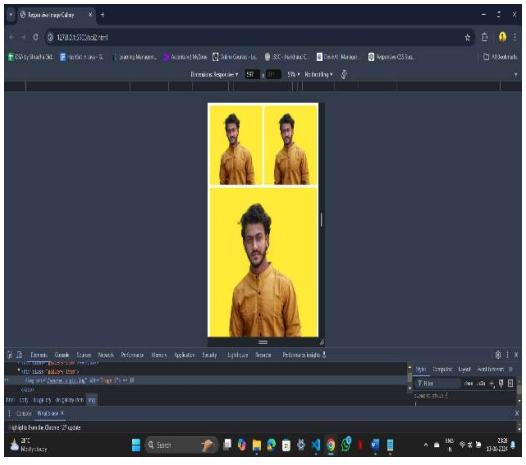
SOL. 2

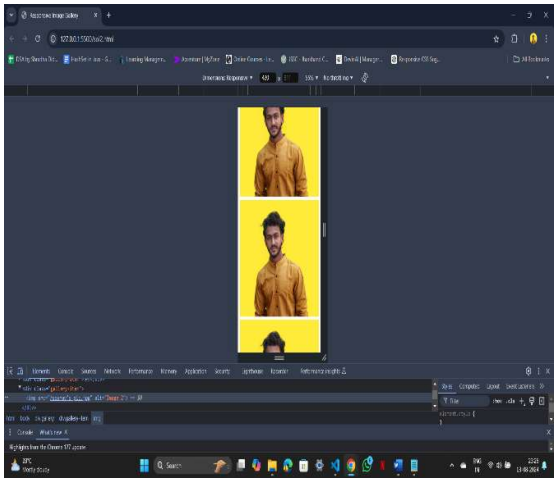
Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Image Gallery</title>
  <style>
    /* Basic styling for the gallery */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    .gallery {
      display: flex;
      flex-wrap: wrap;
      gap: 10px;
      padding: 10px;
    }
    .gallery img {
      width: 100%;
      height: auto;
      display: block;
    }
    .gallery-item {
      flex: 1 1 calc(33.333% - 20px);
      box-sizing: border-box;
```

```
    }
    /* Media query for screens smaller than 800px */
    @media (max-width: 800px) {
      .gallery-item {
        flex: 1 1 calc(50% - 20px);
      }
    }
    /* Media query for screens smaller than 500px */
    @media (max-width: 500px) {
      .gallery-item {
        flex: 1 1 100%;
      }
    }
  </style>
</head>
<body>
  <div class="gallery">
    <div class="gallery-item">
      <img src="/sourav's pic.jpg" alt="Image 1">
    </div>
    <div class="gallery-item">
      <img src="/sourav's pic.jpg" alt="Image 2">
    </div>
    <div class="gallery-item">
      <img src="/sourav's pic.jpg" alt="Image 3">
    </div>
  </div>
</body>
</html>
```





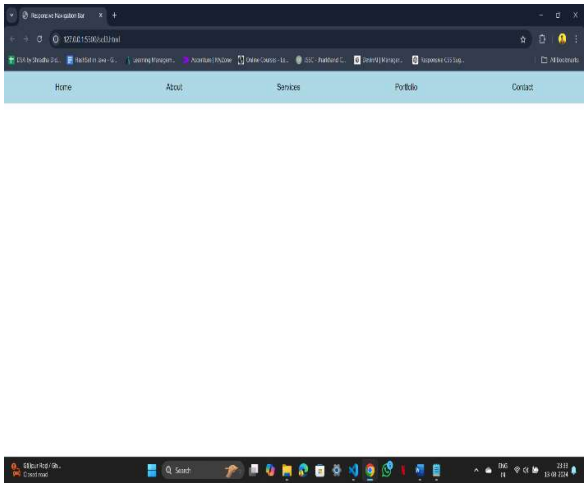


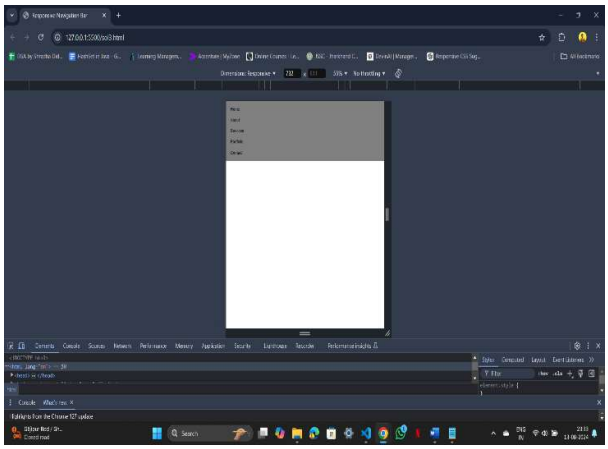
SOL. 3

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Responsive Navigation Bar</title>
  <style>
    /* Basic styling for the navigation bar */
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    nav {
      background-color: lightblue;
      padding: 10px;
    }
    nav ul {
      list-style-type: none;
      margin: 0;
      padding: 0;
      display: flex;
      justify-content: space-around;
    }
    nav ul li {
```

```
      margin: 0;
    }
    nav ul li a {
      text-decoration: none;
      color: black;
      padding: 10px 20px;
      display: block;
    }
    /* Media query for screens smaller than 768px */
    @media (max-width: 768px) {
      nav {
        background-color: gray;
      }
      nav ul {
        flex-direction: column;
        align-items: flex-start;
      }

      nav ul li {
        width: 100%;
      }
      nav ul li a {
        padding: 10px;
        width: 100%;
      }
    }
  </style>
</head>
<body>
  <nav>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#about">About</a></li>
      <li><a href="#services">Services</a></li>
      <li><a href="#portfolio">Portfolio</a></li>
      <li><a href="#contact">Contact</a></li>
    </ul>
  </nav>
</body>
</html>
```

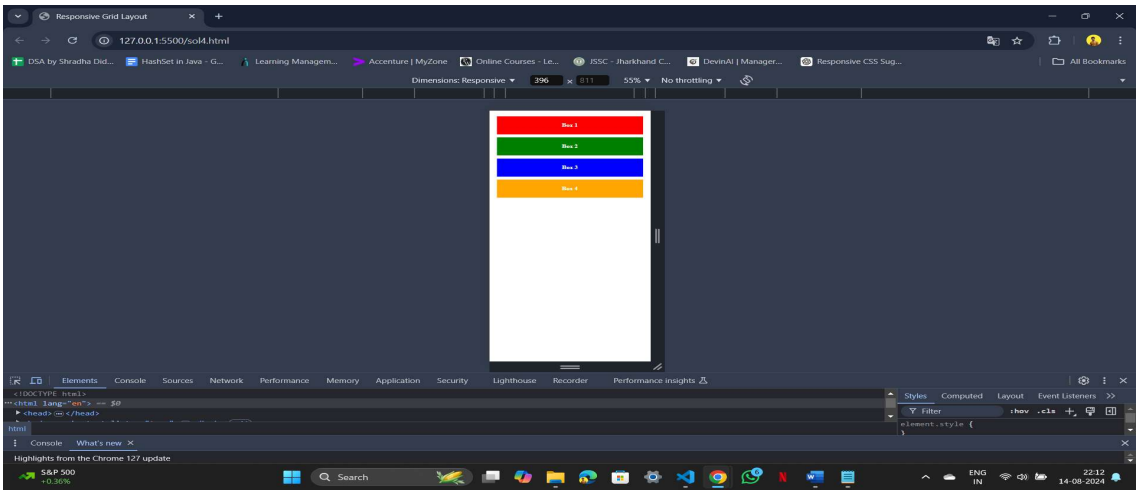




SOL. 4

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Grid Layout</title>
    <style>
        body {
            display: grid;
            grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
            gap: 10px;
            padding: 10px;
        }
        .grid-item {
            padding: 20px;
            text-align: center;
            color: white;
            font-weight: bold;
        }
        .box1 {
```

```
        background-color: red;
    }
    .box2 {
        background-color: green;
    }
    .box3 {
        background-color: blue;
    }

    .box4 {
        background-color: orange;
    }
  </style>
</head>
<body>
  <div class="grid-item box1">Box 1</div>
  <div class="grid-item box2">Box 2</div>
  <div class="grid-item box3">Box 3</div>
  <div class="grid-item box4">Box 4</div>
</body>
</html>
```



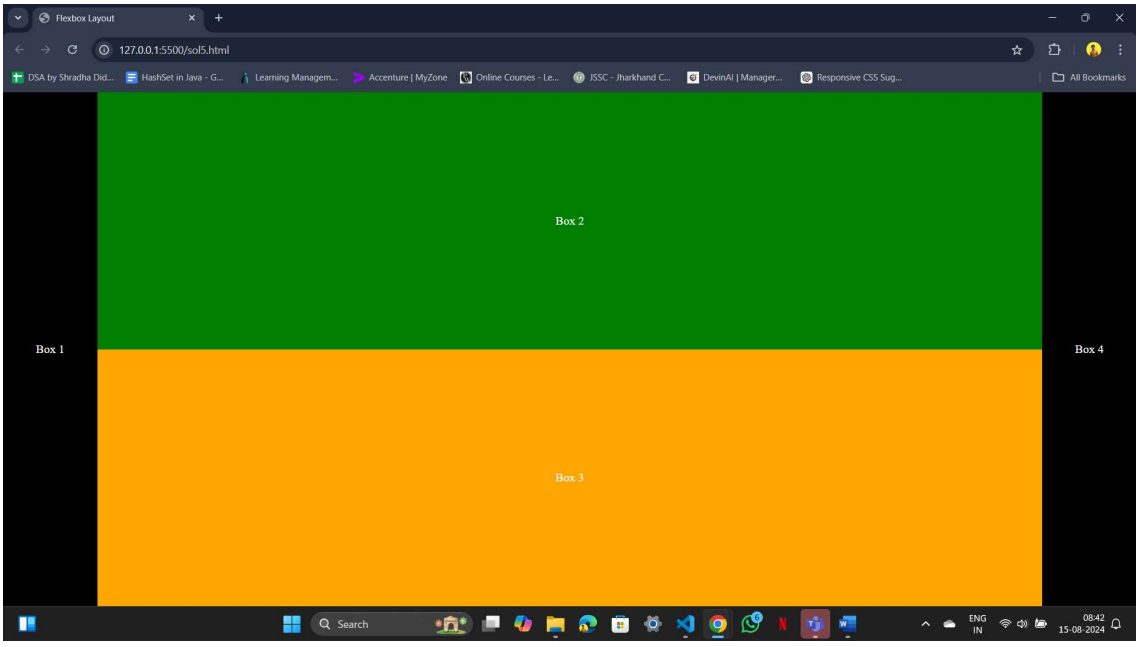
SOL. 5

Index.html

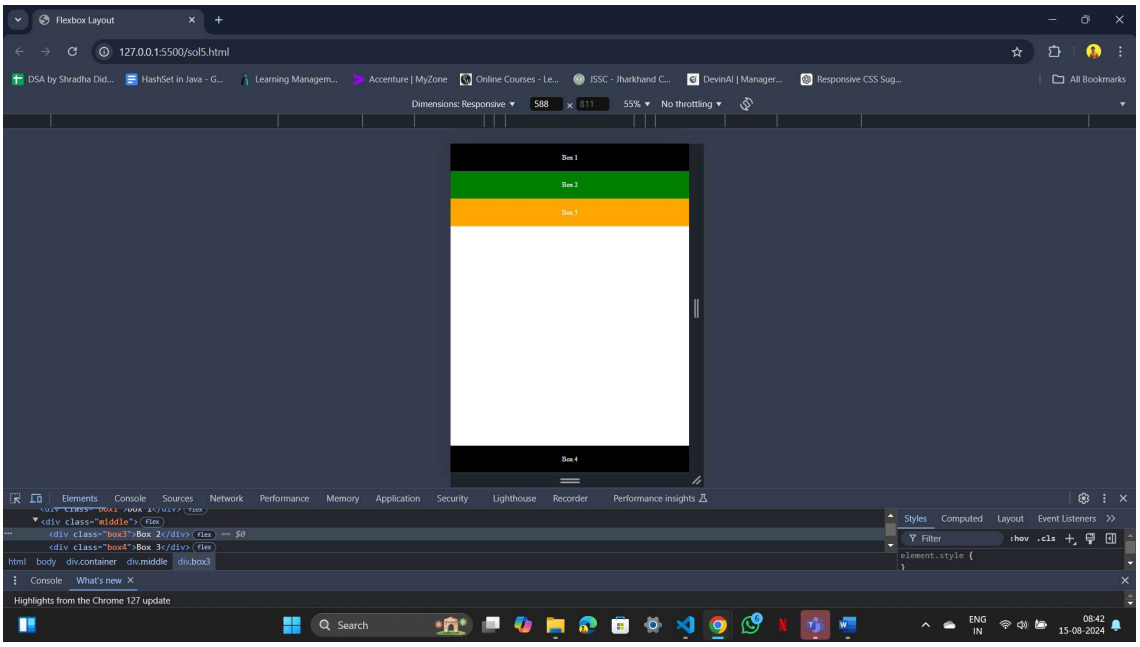
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flexbox Layout</title>
    <style>
        body {
            margin: 0;
            display: flex;
            height: 100vh;
            flex-direction: column;
        }
        .container {
            display: flex;
            flex: 1;
        }
        .box1,
        .box2 {
            flex: 0.1;
```

```
    background-color: black;
    color: white;
    display: flex;
    align-items: center;
    justify-content: center;
}

.middle {
    display: flex;
    flex-direction: column;
    flex: 1;
}
.box3,
.box4 {
    flex: 1;
    display: flex;
    align-items: center;
    justify-content: center;
    color: white;
}
.box3 {
    background-color: green;
}
.box4 {
    background-color: orange;
}
@media (max-width: 768px) {
    .container {
        flex-direction: column;
    }
    .box3,
    .box4 {
```

```
            height: auto;
            flex: 0.1;  }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box1">Box 1</div>
        <div class="middle">
            <div class="box3">Box 2</div>
            <div class="box4">Box 3</div>
        </div>
        <div class="box2">Box 4</div>
    </div>
</body>
</html>
```



PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0



Tailwind CSS

SOL. 1

User login page interface using Tailwind CSS with Vite

Step 1: Set Up a New Vite Project

1. **Initialize Vite:** Open your terminal and run the following command to create a new Vite project:

npm create vite@latest my-login-page

2. **Navigate to the Project Directory:**

cd my-login-page

3. **Install Dependencies:** Install the necessary dependencies for Tailwind CSS:

npm install -D tailwindcss postcss autoprefixer

4. **Initialize Tailwind CSS:** Initialize Tailwind CSS by running the following command:

npx tailwindcss init -p

This command will generate a tailwind.config.js and a postcss.config.js file in your project.

Step 2: Configure Tailwind CSS

- **Configure tailwind.config.js:** Open the tailwind.config.js file and replace the content inside content with the following:

  /** @type {import('tailwindcss').Config} */

  module.exports = {

   content: [

```
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

- **Create Tailwind CSS Entry Point:** In the `src` directory, create a new CSS file (e.g., `styles.css`), and add the following lines:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

- **Include the CSS in Your Project:** Open `main.js` or `main.jsx` (depending on your setup) and import the CSS file:

```
import './styles.css';
```

Step 3: Create the HTML Structure

- **Update `index.html`:** Replace the content of your `index.html` file with the following structure:

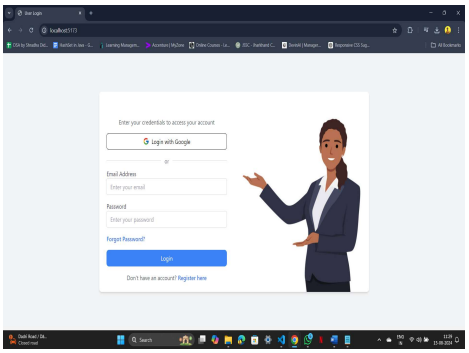
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>User Login</title>
  <link rel="stylesheet" href="styles.css">
  <script type="module" src="/src/main.tsx"></script>
</head>

<body class="bg-gray-100 flex items-center justify-center min-h-screen
p-4">
  <div class="w-full max-w-md md:max-w-4xl bg-white rounded-lg shadow-
md flex flex-col md:flex-row">

    <!-- Left Column: Login Form -->
    <div class="w-full p-6 md:w-1/2">
      <h2 class="text-2xl font-bold text-center mb-2">Login</h2>
```

```
      <p class="text-gray-600 text-center mb-4">Enter your credentials
to access your account</p>

      <!-- Google Login Button -->
      <button
        class="w-full bg-white border border-black text-black py-2
rounded-lg flex items-center justify-center hover:bg-gray-100
focus:outline-none focus:ring-2 focus:ring-gray-300 mb-4">
        <img src="./src/assets/icons8-google-48.png" alt="Google"
class="w-5 h-5 mr-2">
        Login with Google
      </button>

      <div class="flex items-center my-2">
        <hr class="w-full border-gray-300">
        <span class="px-3 text-gray-500">or</span>
        <hr class="w-full border-gray-300">
      </div>

      <form class="space-y-4">
        <div>
          <label class="block text-gray-700">Email Address</label>
          <input type="email"
            class="w-full bg-white text-gray-700 px-3 py-2 border
border-gray-300 rounded focus:outline-none focus:ring focus:border-
blue-500"
            placeholder="Enter your email" required>
        </div>

        <div>
          <label class="block text-gray-700">Password</label>
          <input type="password"
            class="w-full bg-white text-gray-700 px-3 py-2 border
border-gray-300 rounded focus:outline-none focus:ring focus:border-
blue-500"
            placeholder="Enter your password" required>
        </div>

        <div class="flex justify-between">
          <a href="#" class="text-blue-500 hover:underline">Forgot
Password?</a>
        </div>

        <button type="submit"
          class="w-full bg-blue-500 text-white py-2 rounded-lg
hover:bg-blue-600 focus:outline-none focus:ring-2 focus:ring-blue-
500">Login</button>
      </form>

      <p class="text-center text-gray-600 mt-4">
```

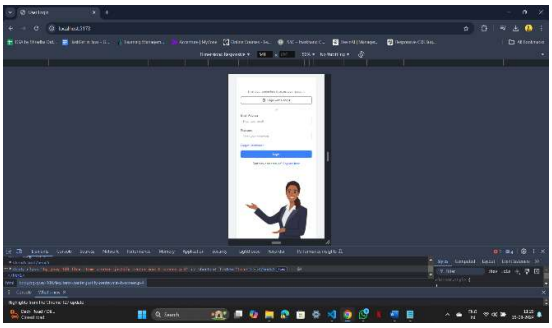
```
        Don't have an account? <a href="#" class="text-blue-500
hover:underline">Register here</a>
      </p>
    </div>

    <!-- Right Column: Full-Height Image -->
    <div class="w-full md:w-1/2 flex items-center justify-center">
      <img src="./src/assets/image-girl.png" alt="Girl"
        class="w-full h-auto md:h-full object-cover rounded-b-lg
md:rounded-r-lg">
    </div>
  </div>
</body>

</html>
```

OUTPUT:

Large screen:



Small screen:



SOL. 2

Image gallery interface using Tailwind CSS with Vite

Step 1: Set Up a New Vite Project

5. **Initialize Vite:** Open your terminal and run the following command to create a new Vite project:

npm create vite@latest my-image-gallery

6. **Navigate to the Project Directory:**

cd my-image-gallery

7. **Install Dependencies:** Install the necessary dependencies for Tailwind CSS:

npm install -D tailwindcss postcss autoprefixer

8. **Initialize Tailwind CSS:** Initialize Tailwind CSS by running the following command:

npx tailwindcss init -p

This command will generate a tailwind.config.js and a postcss.config.js file in your project.

Step 2: Configure Tailwind CSS

- **Configure tailwind.config.js:** Open the tailwind.config.js file and replace the content inside content with the following:

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './index.css'
createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

- **Create Tailwind CSS Entry Point:** In the `src` directory, create a new CSS file (e.g., `styles.css`), and add the following lines:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

- **Include the CSS in Your Project:** Open `main.js` or `main.jsx` (depending on your setup) and import the CSS file:

```
import './styles.css';
```

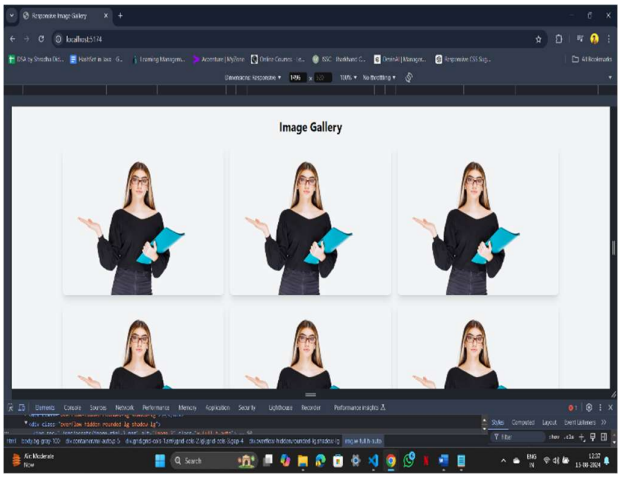
Step 3: Create the HTML Structure

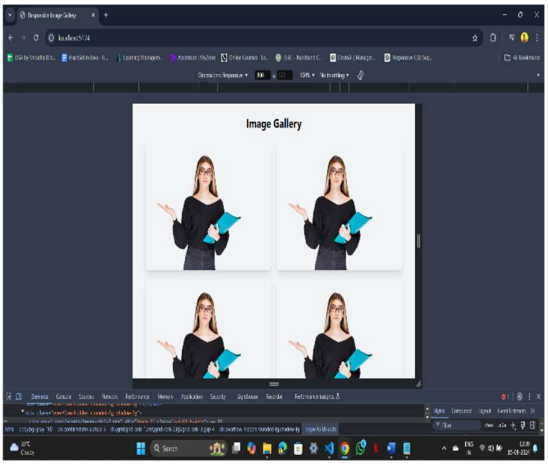
- **Update `index.html`:** Replace the content of your `index.html` file with the following structure:

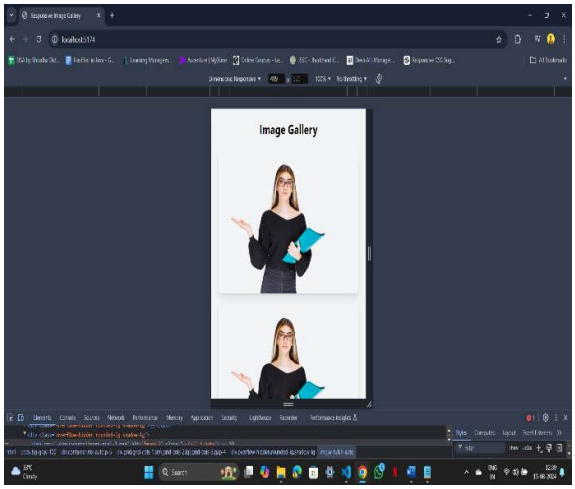
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Responsive Image Gallery</title>
```

```
  <link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.mi
n.css" rel="stylesheet">
</head>
<body class="bg-gray-100">
  <script src="./src/main.jsx"></script>
  <div class="container mx-auto p-5">
    <h1 class="text-2xl font-bold text-center mb-5">Image Gallery</h1>
    <div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
      <div class="overflow-hidden rounded-lg shadow-lg">
        <img src="./src/assets/image-girl-2.png" alt="Image 1"
class="w-full h-auto">
      </div>
      <div class="overflow-hidden rounded-lg shadow-lg">
        <img src="./src/assets/image-girl-2.png" alt="Image 2"
class="w-full h-auto">
      </div>
      <div class="overflow-hidden rounded-lg shadow-lg">
        <img src="./src/assets/image-girl-2.png" alt="Image 3"
class="w-full h-auto">
      </div>
      <div class="overflow-hidden rounded-lg shadow-lg">
        <img src="./src/assets/image-girl-2.png" alt="Image 4"
class="w-full h-auto">
      </div>
      <div class="overflow-hidden rounded-lg shadow-lg">
        <img src="./src/assets/image-girl-2.png" alt="Image 5"
class="w-full h-auto">
      </div>
      <div class="overflow-hidden rounded-lg shadow-lg">
        <img src="./src/assets/image-girl-2.png" alt="Image 6"
class="w-full h-auto">
      </div>
    </div>
  </div>
</body>
</html>
```

OUTPUT:







SOL. 3

Dashboard using Tailwind CSS with Vite

Step 1: Set Up a New Vite Project

9. **Initialize Vite:** Open your terminal and run the following command to create a new Vite project:

npm create vite@latest my-dashboard

10. **Navigate to the Project Directory:**

cd my-dashboard

11. **Install Dependencies:** Install the necessary dependencies for Tailwind CSS:

npm install -D tailwindcss postcss autoprefixer

12. **Initialize Tailwind CSS:** Initialize Tailwind CSS by running the following command:

npx tailwindcss init -p

This command will generate a tailwind.config.js and a postcss.config.js file in your project.

Step 2: Configure Tailwind CSS

- **Configure tailwind.config.js:** Open the tailwind.config.js file and replace the content inside content with the following:

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import App from './App.jsx'
import './index.css'
createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

- **Create Tailwind CSS Entry Point:** In the `src` directory, create a new CSS file (e.g., `styles.css`), and add the following lines:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

- **Include the CSS in Your Project:** Open `main.js` or `main.jsx` (depending on your setup) and import the CSS file:

```
import './styles.css';
```

Step 3: Create the HTML Structure

- **Update `index.html`:** Replace the content of your `index.html` file with the following structure:

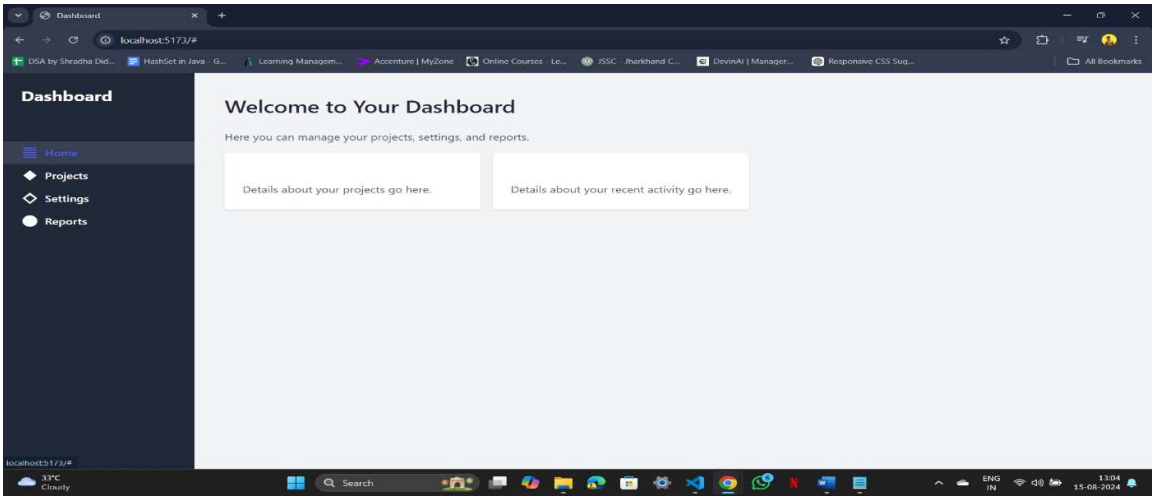
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link href="/style.css" rel="stylesheet">
  <title>Dashboard</title>
</head>

<body class="bg-gray-100">
```

```
  <script type="module" src="/src/main.jsx"></script>
  <div class="flex h-screen">
    <!-- Sidebar -->
    <aside class="w-64 bg-gray-800 text-white">
      <div class="p-6">
        <h1 class="text-2xl font-bold">Dashboard</h1>
      </div>
      <nav class="mt-10">
        <ul>
          <li class="px-6 py-2 hover:bg-gray-700">
            <a href="#" class="flex items-center">
              <svg class="w-6 h-6 mr-2" fill="currentColor" viewBox="0
0 24 24">
                <path d="M3 3h18v2H3V3zm0 4h18v2H3V7zm0
4h18v2H3v-2zm0 4h18v2H3v-2z" />
              </svg>
              <span>Home</span>
            </a>
          </li>
          <li class="px-6 py-2 hover:bg-gray-700">
            <a href="#" class="flex items-center">
              <svg class="w-6 h-6 mr-2" fill="currentColor" viewBox="0
0 24 24">
                <path d="M12 3l9 9-9 9-9-9 9-9z" />
              </svg>
              <span>Projects</span>
            </a>
          </li>
          <li class="px-6 py-2 hover:bg-gray-700">
            <a href="#" class="flex items-center">
              <svg class="w-6 h-6 mr-2" fill="currentColor" viewBox="0
0 24 24">
                <path d="M12 2l10 10-10 10-10-10L12 2zm0 4.5L5.5 12 12
18.5 18.5 12 12 6.5z" />
              </svg>
              <span>Settings</span>
            </a>
          </li>
          <li class="px-6 py-2 hover:bg-gray-700">
            <a href="#" class="flex items-center">
              <svg class="w-6 h-6 mr-2" fill="currentColor" viewBox="0
0 24 24">
                <path d="M12 2a10 10 0 100 20 10 10 0 000-20zm0 18a8 8
0 100-16 8 8 0 000 16z" />
              </svg>
              <span>Reports</span>
            </a>
          </li>
        </ul>
      </nav>
    </aside>
```

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

```
    <!-- Main Content Area -->
    <main class="flex-1 p-10">
      <h2 class="text-3xl font-semibold text-gray-800 mb-6">Welcome to
Your Dashboard</h2>
      <p class="text-gray-600 mb-4">Here you can manage your projects,
settings, and reports.</p>

      <div class="grid grid-cols-2 gap-4">
        <div class="bg-white p-6 rounded shadow">
          <h3 class="text-xl font-semibold">Project Overview</h3>
          <p class="text-gray-600">Details about your projects go
here.</p>
        </div>
        <div class="bg-white p-6 rounded shadow">
          <h3 class="text-xl font-semibold">Recent Activity</h3>
          <p class="text-gray-600">Details about your recent activity
go here.</p>
        </div>
      </div>
    </main>
  </div>

</body>
</html>
```

OUTPUT:



PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

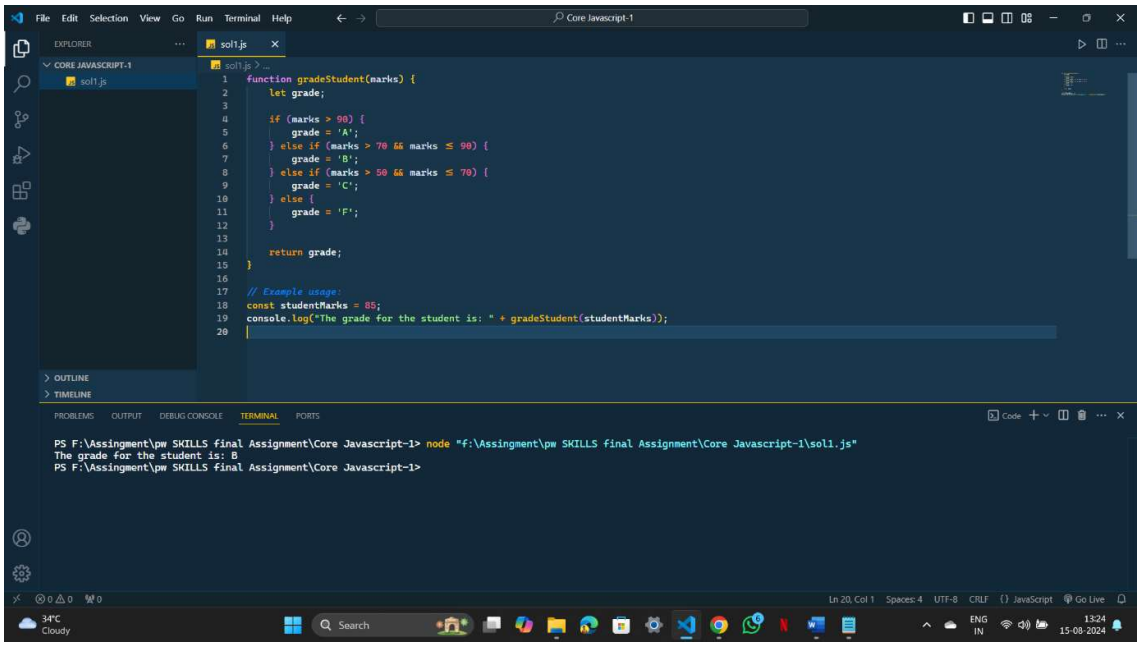
Core Javascript-1

SOL. 2

Index.js

```
function gradeStudent(marks) {
    let grade;

    if (marks > 90) {
        grade = 'A';
    } else if (marks > 70 && marks <= 90) {
        grade = 'B';
    } else if (marks > 50 && marks <= 70) {
        grade = 'C';
    } else {
        grade = 'F';
    }

    return grade;
}

// Example usage:
const studentMarks = 85;
console.log("The grade for the student is: " +
gradeStudent(studentMarks));
```

OUTPUT:



SOL. 3

What Are Loops?

Loops are programming constructs that allow you to execute a block of code repeatedly, either for a specific number of times or until a certain condition is met. They are essential for tasks like iterating over arrays, processing data, or automating repetitive tasks.

Why Do We Need Loops?

Loops help avoid code repetition, making the code more concise, maintainable, and efficient. Instead of writing the same code multiple times, you can write it once and loop through it as many times as needed.

Types of Loops in JavaScript

1. **for Loop**
2. **while Loop**
3. **do...while Loop**
4. **for...in Loop**
5. **for...of Loop**

Let's explore each one with its syntax and examples:

---

1. for Loop

The for loop is used when you know in advance how many times you want to execute a statement or a block of statements.

Syntax:

```
for (initialization; condition; increment/decrement) {
  // Code to be executed
}
```

Example:

```
for (let i = 0; i < 5; i++) {
  console.log("Iteration number: " + i);
}
```

This loop will run 5 times, printing the iteration number from 0 to 4.

---

2. while Loop

The while loop is used when you want to execute a block of code as long as the specified condition is true.

Syntax:

```
while (condition) {
  // Code to be executed
}
```

Example:

```
let i = 0;
while (i < 5) {
  console.log("Iteration number: " + i);
  i++;
}
```

This loop will also run 5 times, printing the iteration number from 0 to 4.

---

3. do...while Loop

The do...while loop is similar to the while loop, but the block of code is executed at least once before the condition is tested.

Syntax:

javascript

Copy code

```
do {
  // Code to be executed
} while (condition);
```

Example:

```
let i = 0;
do {
  console.log("Iteration number: " + i);
  i++;
} while (i < 5);
```

This loop will run 5 times, printing the iteration number from 0 to 4, even if the condition is false at the start.

---

4. for...in Loop

The for...in loop is used to iterate over the properties of an object.

Syntax:

```
for (variable in object) {
  // Code to be executed
}
```

Example:

```
const student = {name: "John", age: 20, grade: "A"};

for (let key in student) {
  console.log(key + ": " + student[key]);
}
```

This loop will print the properties of the student object.

---

5. for...of Loop

The for...of loop is used to iterate over the values of an iterable object, like an array.

Syntax:

```
for (variable of iterable) {
  // Code to be executed
}
```

Example:

```
const numbers = [1, 2, 3, 4, 5];

for (let num of numbers) {
  console.log(num);
}
```

This loop will print each number in the numbers array.

Summary

- **for Loop: Best when you know the exact number of iterations.**
- **while Loop: Best when you don't know how many times you'll need to loop.**
- **do...while Loop: Like while, but guarantees at least one execution.**
- **for...in Loop: Best for iterating over object properties.**

- **for...of Loop: Best for iterating over values of arrays or other iterable objects.**

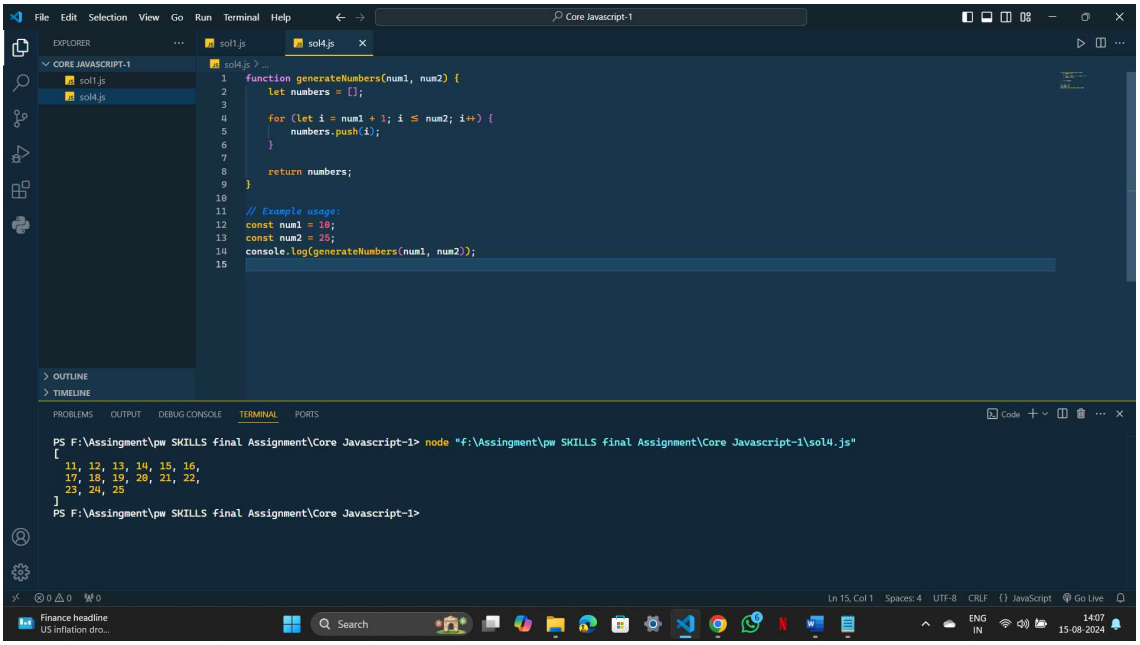
Each loop serves a different purpose and can be chosen based on the specific requirements of your task.

SOL. 4

Index.js

```
function generateNumbers(num1, num2) {
    let numbers = [];

    for (let i = num1 + 1; i <= num2; i++) {
        numbers.push(i);
    }

    return numbers;
}

// Example usage:
const num1 = 10;
const num2 = 25;
console.log(generateNumbers(num1, num2));
```

OUTPUT:



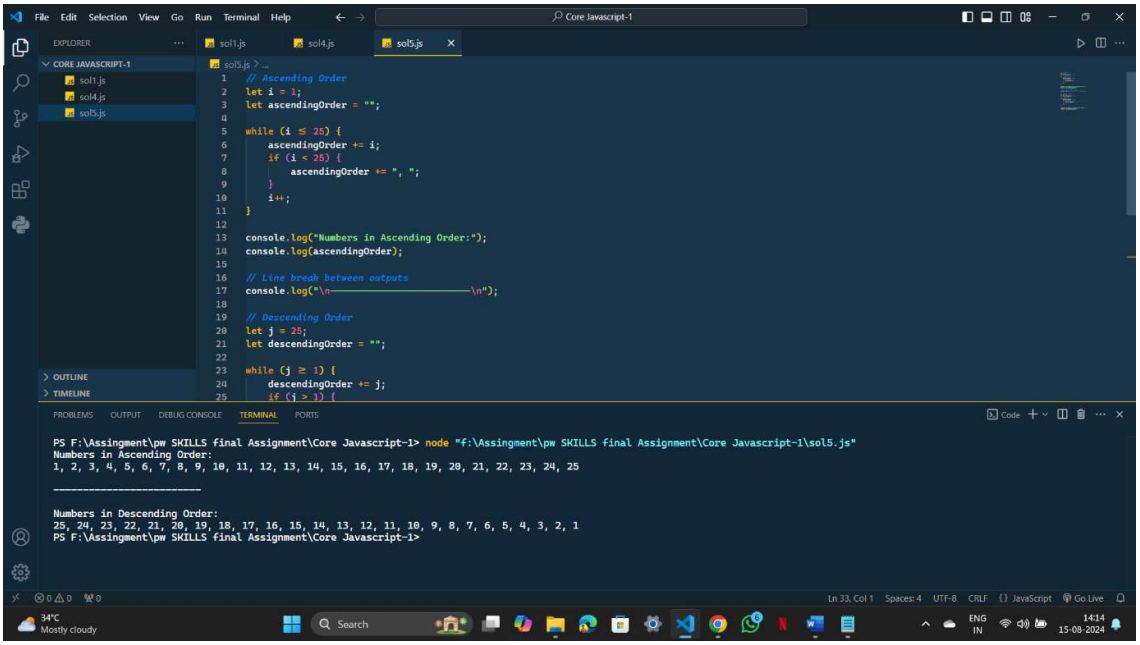
SOL. 5

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

Index.js

```
// Ascending Order
let i = 1;
let ascendingOrder = "";

while (i <= 25) {
    ascendingOrder += i;
    if (i < 25) {
        ascendingOrder += ", ";
    }
    i++;
}

console.log("Numbers in Ascending Order:");
console.log(ascendingOrder);

// Line break between outputs
console.log("\n-------------------------\n");

// Descending Order
let j = 25;
let descendingOrder = "";

while (j >= 1) {
    descendingOrder += j;
    if (j > 1) {
        descendingOrder += ", ";
    }
    j--;
}

console.log("Numbers in Descending Order:");
console.log(descendingOrder);
```

OUTPUT:



Core Javascript-2

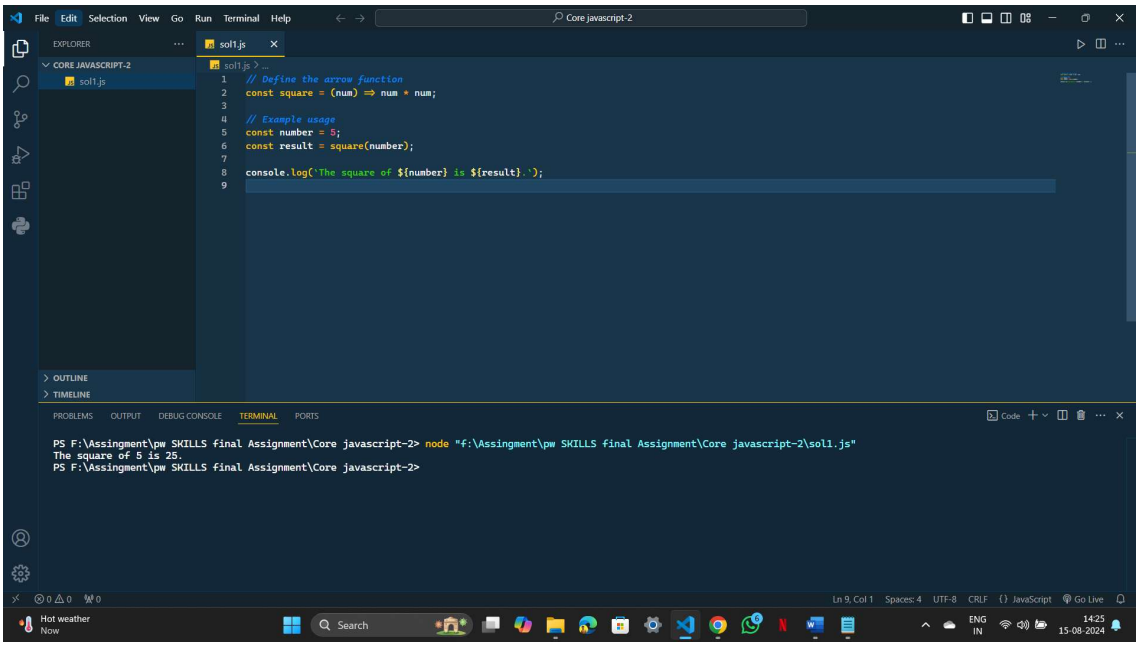
SOL. 1

Index.js

```
const square = (num) => num * num;

// Example usage
const number = 5;
const result = square(number);

console.log(`The square of ${number} is ${result}.`);
```

OUTPUT:

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0



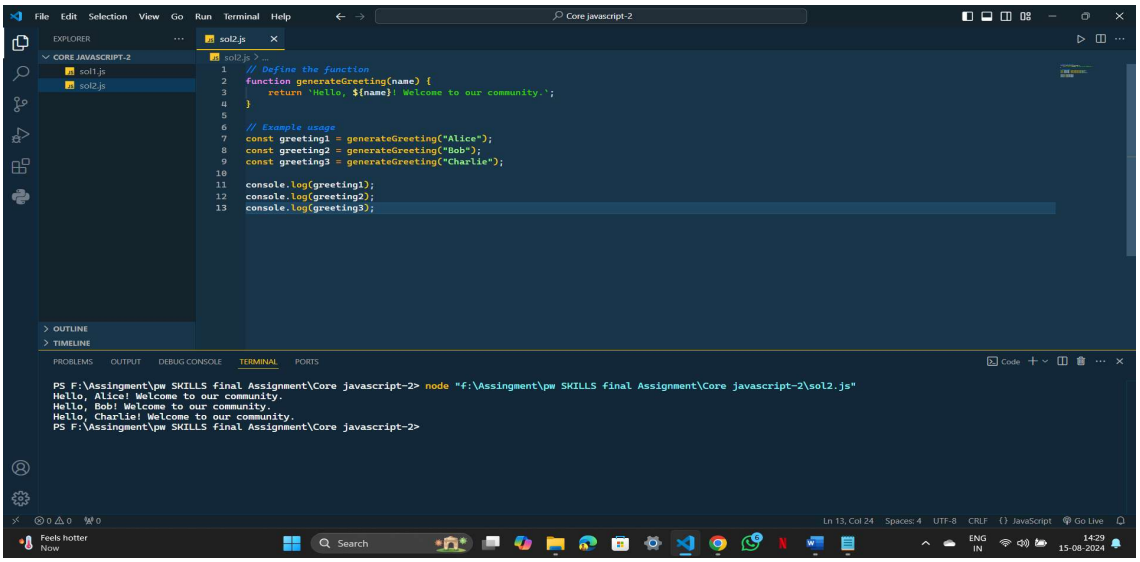
SOL. 2

Index.js

```
// Define the function
function generateGreeting(name) {
    return `Hello, ${name}! Welcome to our community.`;
}

// Example usage
const greeting1 = generateGreeting("Alice");
const greeting2 = generateGreeting("Bob");
const greeting3 = generateGreeting("Charlie");

console.log(greeting1);
console.log(greeting2);
console.log(greeting3);
```

OUTPUT:

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0



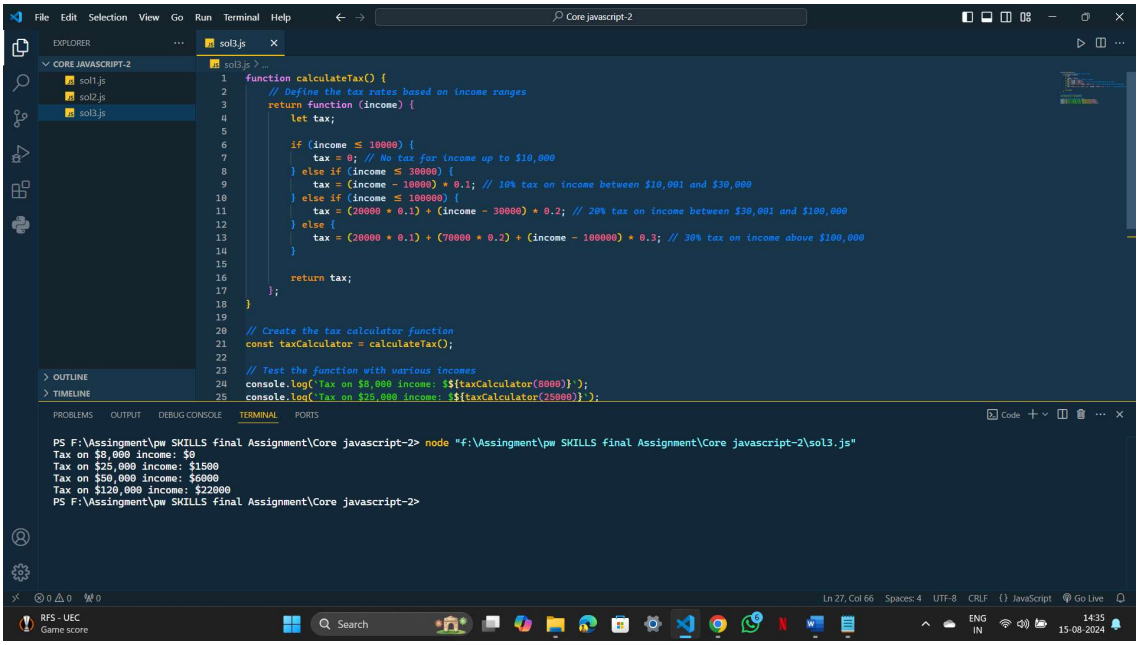
SOL. 3

Index.js

```
function calculateTax() {
    // Define the tax rates based on income ranges
    return function (income) {
        let tax;

        if (income <= 10000) {
            tax = 0; // No tax for income up to $10,000
        } else if (income <= 30000) {
            tax = (income - 10000) * 0.1; // 10% tax on income between
$10,001 and $30,000
        } else if (income <= 100000) {
            tax = (20000 * 0.1) + (income - 30000) * 0.2; // 20% tax
on income between $30,001 and $100,000
        } else {
            tax = (20000 * 0.1) + (70000 * 0.2) + (income - 100000) *
0.3; // 30% tax on income above $100,000
        }

        return tax;
    };
}
// Create the tax calculator function
const taxCalculator = calculateTax();

// Test the function with various incomes
```

```
console.log(`Tax on $8,000 income: $${taxCalculator(8000)}`);
console.log(`Tax on $25,000 income: $${taxCalculator(25000)}`);
console.log(`Tax on $50,000 income: $${taxCalculator(50000)}`);
console.log(`Tax on $120,000 income: $${taxCalculator(120000)}`);
```

OUTPUT:



SOL. 4

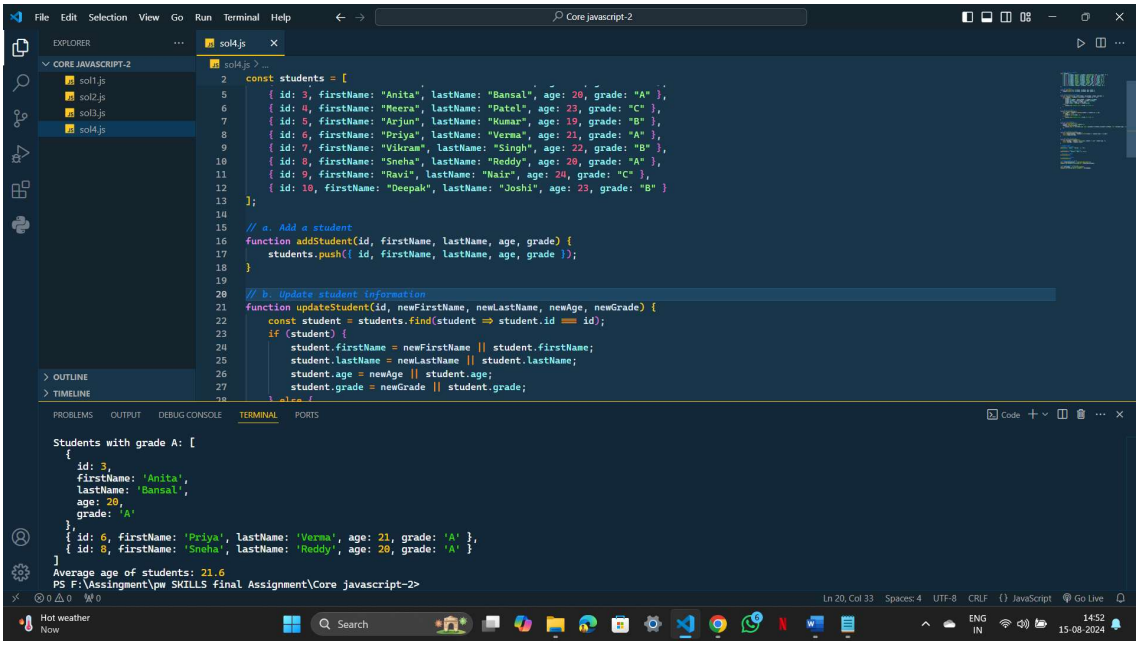
Index.js

```
// Initial array of students
const students = [
    { id: 1, firstName: "Sourav", lastName: "Chakraborty", age: 21,
grade: "A" },
    { id: 2, firstName: "Rahul", lastName: "Sharma", age: 22, grade:
"B" },
    { id: 3, firstName: "Anita", lastName: "Bansal", age: 20, grade:
"A" },
    { id: 4, firstName: "Meera", lastName: "Patel", age: 23, grade:
"C" },
    { id: 5, firstName: "Arjun", lastName: "Kumar", age: 19, grade:
"B" },
    { id: 6, firstName: "Priya", lastName: "Verma", age: 21, grade:
"A" },
    { id: 7, firstName: "Vikram", lastName: "Singh", age: 22, grade:
"B" },
    { id: 8, firstName: "Sneha", lastName: "Reddy", age: 20, grade:
"A" },
```

```
    { id: 9, firstName: "Ravi", lastName: "Nair", age: 24, grade: "C"
},
    { id: 10, firstName: "Deepak", lastName: "Joshi", age: 23, grade:
"B" }
];

// a. Add a student
function addStudent(id, firstName, lastName, age, grade) {
    students.push({ id, firstName, lastName, age, grade });
}

// b. Update student information
function updateStudent(id, newFirstName, newLastName, newAge,
newGrade) {
    const student = students.find(student => student.id === id);
    if (student) {
        student.firstName = newFirstName || student.firstName;
        student.lastName = newLastName || student.lastName;
        student.age = newAge || student.age;
        student.grade = newGrade || student.grade;
    } else {
        console.log(`Student with id ${id} not found.`);
    }
}

// c. Delete a student
function deleteStudent(id) {
    const index = students.findIndex(student => student.id === id);
    if (index !== -1) {
        students.splice(index, 1);
    } else {
        console.log(`Student with id ${id} not found.`);
    }
}

// d. List all students
function listAllStudents() {
    console.log("List of all students:");
    students.forEach(student => {
        console.log(`ID: ${student.id}, Name: ${student.firstName}
${student.lastName}, Age: ${student.age}, Grade: ${student.grade}`);
    });
}

// e. Find students by grade
function findStudentsByGrade(grade) {
    const studentsInGrade = students.filter(student => student.grade
=== grade);
    return studentsInGrade;
}
```

```
// f. Calculate average age
function calculateAverageAge() {
    const totalAge = students.reduce((sum, student) => sum +
student.age, 0);
    return totalAge / students.length;
}

// Example usage:
// Adding a new student
addStudent(2, "Rahul", "Sharma", 22, "B");

// Updating a student's information
updateStudent(1, "Sourav", "Roy", 22, "A+");

// Deleting a student
deleteStudent(2);

// Listing all students
listAllStudents();

// Finding students by grade
const studentsWithGradeA = findStudentsByGrade("A");
console.log("Students with grade A:", studentsWithGradeA);

// Calculating the average age
const averageAge = calculateAverageAge();
console.log("Average age of students:", averageAge);
```

OUTPUT:



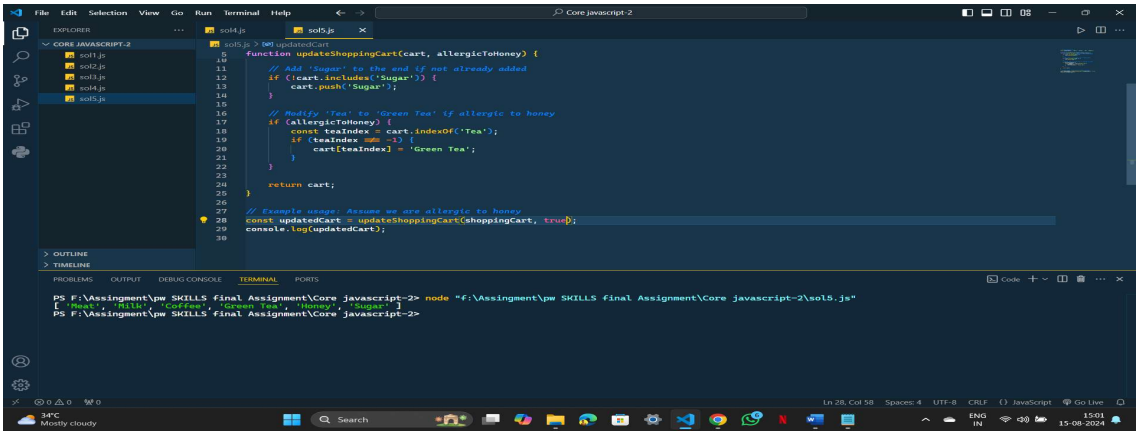
SOL. 5

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)
Decode full stack web dev 1.0

Index.html

```javascript
// Initial shopping cart
const shoppingCart = ['Milk', 'Coffee', 'Tea', 'Honey'];

// Function to update the shopping cart
function updateShoppingCart(cart, allergicToHoney) {
    // Add 'Meat' to the beginning if not already added
    if (!cart.includes('Meat')) {
        cart.unshift('Meat');
    }

    // Add 'Sugar' to the end if not already added
    if (!cart.includes('Sugar')) {
        cart.push('Sugar');
    }

    // Modify 'Tea' to 'Green Tea' if allergic to honey
    if (allergicToHoney) {
        const teaIndex = cart.indexOf('Tea');
        if (teaIndex !== -1) {
            cart[teaIndex] = 'Green Tea';
        }
    }

    return cart;
}

// Example usage: Assume we are allergic to honey
const updatedCart = updateShoppingCart(shoppingCart, true);
console.log(updatedCart);
```

OUTPUT:



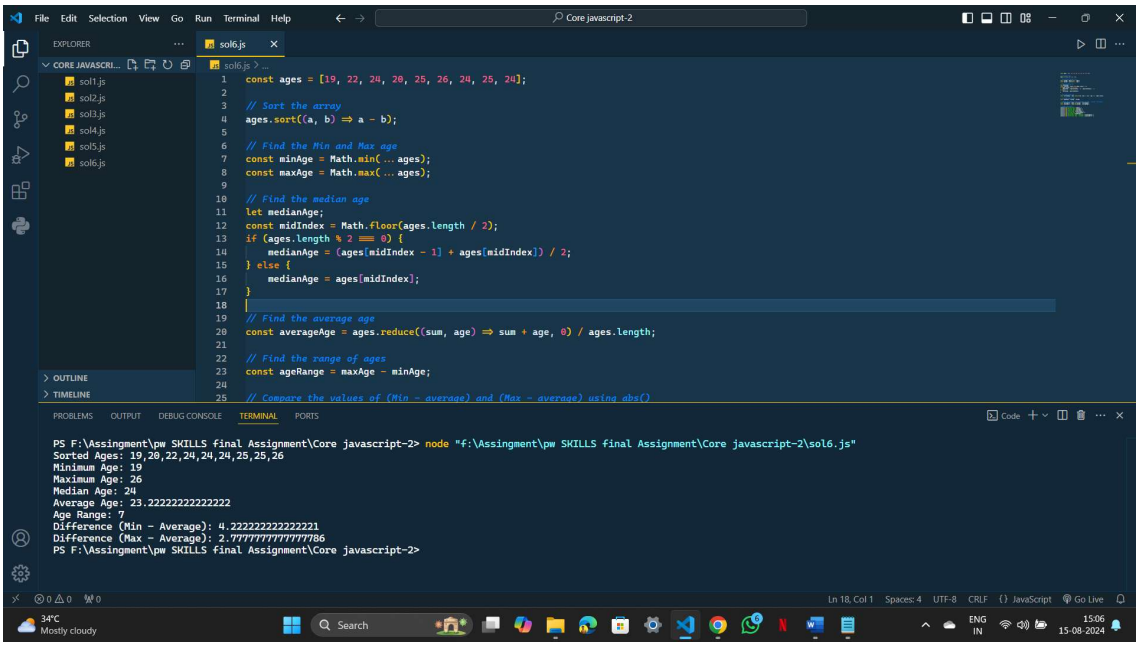
Modify the allergicToHoney variable to true or false based on whether you're allergic to honey.

SOL. 6

Index.js

```
const ages = [19, 22, 24, 20, 25, 26, 24, 25, 24];

// Sort the array
ages.sort((a, b) => a - b);

// Find the Min and Max age
const minAge = Math.min(...ages);
const maxAge = Math.max(...ages);

// Find the median age
let medianAge;
const midIndex = Math.floor(ages.length / 2);
if (ages.length % 2 === 0) {
    medianAge = (ages[midIndex - 1] + ages[midIndex]) / 2;
} else {
    medianAge = ages[midIndex];
}

// Find the average age
const averageAge = ages.reduce((sum, age) => sum + age, 0) /
ages.length;

// Find the range of ages
const ageRange = maxAge - minAge;

// Compare the values of (Min - average) and (Max - average) using
abs()
const minAvgDiff = Math.abs(minAge - averageAge);
const maxAvgDiff = Math.abs(maxAge - averageAge);

// Output the results
console.log(`Sorted Ages: ${ages}`);
console.log(`Minimum Age: ${minAge}`);
console.log(`Maximum Age: ${maxAge}`);
console.log(`Median Age: ${medianAge}`);
console.log(`Average Age: ${averageAge}`);
console.log(`Age Range: ${ageRange}`);
console.log(`Difference (Min - Average): ${minAvgDiff}`);
console.log(`Difference (Max - Average): ${maxAvgDiff}`);
```

OUTPUT:



SOL. 7

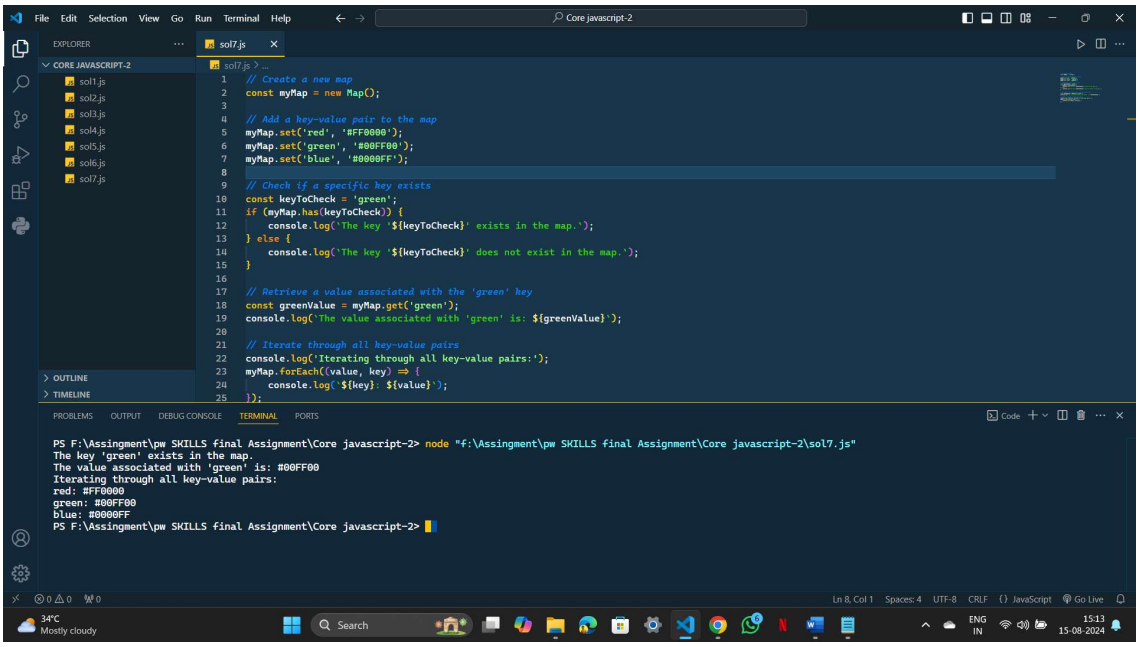
Index.js

```
// Create a new map
const myMap = new Map();

// Add a key-value pair to the map
myMap.set('red', '#FF0000');
myMap.set('green', '#00FF00');
myMap.set('blue', '#0000FF');

// Check if a specific key exists
const keyToCheck = 'green';
if (myMap.has(keyToCheck)) {
    console.log(`The key '${keyToCheck}' exists in the map.`);
} else {
    console.log(`The key '${keyToCheck}' does not exist in the map.`);
}
// Retrieve a value associated with the 'green' key
const greenValue = myMap.get('green');
console.log(`The value associated with 'green' is: ${greenValue}`);

// Iterate through all key-value pairs
console.log('Iterating through all key-value pairs:');
myMap.forEach((value, key) => {
    console.log(`${key}: ${value}`);
});
```

OUTPUT:

PW SKILLS FINAL ASSIGNMENT (FULL STACK WEB DEVELOPMENT)

Decode full stack web dev 1.0



ASSIGNMENT MADE BY SOURAV CHAKRABORTY (souravchh456@gmail.com)