

# Assignment Questions

## Full stack Development



## Getting Started With HTML

1. Write a simple program in HTML that displays the heading "HTML defines the content and structure of your website" on the web browser.
2. Explain the purpose of comments in HTML and provide an example of how to use comments in an HTML document.
3. Write an HTML program that includes a heading, a paragraph of text, a horizontal line, and a line break. Arrange these elements to create a simple web page layout.
4. Write a short note on Tag and element with an example.
5. What is the DOCTYPE Declaration in HTML?

### Core Html

1. Build a simple webpage that displays text as shown in the below image.

This text will be bolded.

*This text will be italic.*

This text will be underlined

This text will be highlighted

This is normal text This will be super scripted This is normal again

This is normal text This text will be subscripted

Normal Text Smal Text

~~This text will be deleted~~

2. Build a simple webpage that helps users navigate different web development-related websites. Note: On clicking the hyperlink the web pages should open in a new tab. Below is a reference image.

### Navigate Me:

Take me to [PW Skills](#) to buy a course.

Take me to [MDN docs](#) to know more about Web Development.

Take me to [PW Skills Lab](#) to practice live coding.

3. Build a simple blog web page with 3 pages home, web development, and web design. Each page must contain hyperlinks to other pages in the top, a heading of the page topic and a paragraph of information. For the home page you can add some information about yourself.
4. Create an ordered list of HTML tags. Each list item must include the tag name and some information about the tag.
6. create a description list of full stack web development tech stack, using the <dl> tag. Each term should be a tech stack name and each description should be a brief explanation of what the tech stack is used for.
7. Create an ordered list of the full stack web development tech stack HTML, CSS, and JS. For each tech stack, create a table that lists the tech stack name, its primary use cases, and some key features or benefits. Below is a reference image.

Eg.

1. HTML	
Primary Use Cases	Key Features/Benefits
Building the structure of web pages	<ul style="list-style-type: none"><li>Simple and easy to learn</li><li>Compatible with all web browsers</li><li>Allows for semantic markup</li></ul>
2. CSS	
Primary Use Cases	Key Features/Benefits
Styling and layout of web pages	<ul style="list-style-type: none"><li>Allows for separation of content and presentation</li><li>Enables responsive design</li><li>Offers a wide range of styling options</li></ul>

7. Build a complex nested list structure representing a multi-level table of contents. Use unordered lists (<ul>) and list items (<li>) with inline-block styling to create a structured layout. Apply formatting tags to enhance the presentation of list items.

Output should look like this:

### Table of Contents

- Part 1: Introduction
- Part 2: Getting Started
  - 2.1 Installing the Software
  - 2.2 Creating a New Project
    - 2.2.1 Project Templates
    - 2.2.2 Customizing Settings
  - 2.3 Exploring the Interface
    - 2.3.1 Toolbar Features
    - 2.3.2 Panel Layout
      - 2.3.2.1 Docking Panels
      - 2.3.2.2 Tabbed Interface
- Part 3: Advanced Topics
  - 3.1 Working with Plugins
    - 3.1.1 Installing Plugins
    - 3.1.2 Plugin Configuration
  - 3.2 Customizing the UI
    - 3.2.1 Changing Themes
    - 3.2.2 Configuring Shortcuts
  - 3.3 Optimizing Performance
    - 3.3.1 Caching Strategies
    - 3.3.2 Resource Minification
- Part 4: Conclusion



8. Create a table to display a conference schedule. Each row corresponds to a time slot, and each column corresponds to a room. Some time slots might have multiple sessions running simultaneously in different rooms. Utilize rowspan and colspan attributes as necessary to accommodate this complex schedule. (use table attribute "cellpadding" to give extra padding in each table cell ).

Output should look like this:

Conference Schedule				
Time	Room 1	Room 2	Room 3	Room 4
9:00 AM - 10:00 AM	Keynote	Session A	Session B	Session C
		Session D	Session E	
	10:30 AM - 11:30 AM	Session F		
12:00 PM - 1:00 PM	Lunch Break			
1:00 PM - 2:00 PM	Session G	Session H	Session I	Session J
	Session K		Session L	Session M

## Media and Forms

1. Create an image gallery that holds multiple images
2. Use video and audio tags to display video and audio with the playback, audio control
3. Modify the previous assignment so that the audio and video play automatically as the page is loaded and they should play infinitely
4. Use iframe to embed the PhysicsWallah Wikipedia page properly
5. Create a sign-up and sign-in form with proper validation
  - a. Sign up form should have a first name, last name, email, password, confirm password, age, gender, and agree to terms and conditions fields at minimum (You can add any other if you like)
  - b. Sign in form should have email, password fields

**NOTE:** Validation is a must.

**Ex:** First name, last name, email cannot be less than 3 characters, age cannot be negative and cannot be greater than 150, fields are required, etc.,

## Starting with CSS

1. Create a simple page with some div tags and show different ways to add CSS as well as what happens when you target the same elements with inline, internal, and external CSS. Also, utilize comments in the project where required.
2. Build an HTML page with multiple paragraphs, each assigned a unique class name. Write CSS rules using class selectors to apply distinct styling to each paragraph. Follow the BEM naming convention and explain how you've named the classes.
3. Develop an HTML form with various input elements. Use CSS to style the form, including setting background colors for input fields. Create a custom color palette for the form elements, and demonstrate how to apply opacity to one of the form sections

## More on CSS

1. Create a navigation bar similar to the below-mentioned image. The navigation bar must contain 5 navigation links that are properly placed within the navigation bar using the CSS box model. The navigation items must change their text color on hover.



2. Create a div centered with an image and paragraph similar to the image mentioned below.



3. Create a navigation bar similar to question 1. For each navigation item, use a different border style.
4. Create a simple webpage with an image and make the image circular using border-radius.
5. Create a simple blog website about Google fonts. The webpage must have the heading with "Oswald" font align center to the page, below the heading have an image of Google fonts and a paragraph about Google fonts in font "Montserrat" paragraph font style should be "montserrat" font.
6. Create a simple website as mentioned in the below image. You can get the assets by visiting <https://pwwskills.com/about-us>.



## About Us

PW Skills' mission is to permeate through every student/professional's outlook towards jobs and change their attitude and perspective from "How Can I Do it?" to "Of Course I Can Do it!". We aim to do this by providing exceptional up skilling courses at affordable rates, while being tech-forward so anyone, anywhere can access and improve their ability to be successful in life.



7. Create a simple webpage with a card similar to question number 2. The card must be semi-transparent as default and on hovering the card must be displayed clearly. Use CSS opacity to achieve this output.

Without hover



After hover



8. Create a simple list of items as mentioned in the below image.

## Requirements

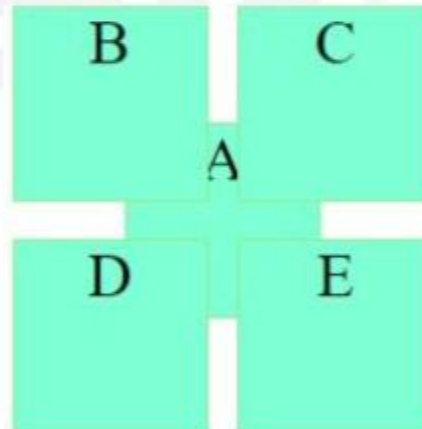
- i. System with minimum i3 processor or better.
- ii. At least 4 GB of RAM.
- iii. Working internet connection.
- iv. Dedication to learn

## Positions in CSS

1. Write code to position 5 equal-sized (50 X 50px) boxes A, B, C, D, and E as follows,

- a. box A 200px from the left, and 200px from the top of the viewport.
- b. box B -30px left, and -30px above from the center of box A.
- c. box C -30px right, and -30px above from the center of box A.
- d. box D -30px left, and -30px below from the center of box A.
- e. box E -30px right, and -30px below from the center of box A.

Expected Output



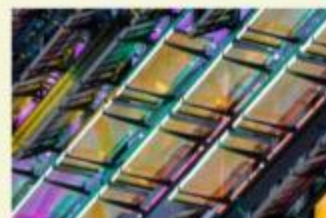
2. Explain the difference between Absolute and Relative positioning.

3. Create a card as shown in the picture below. (You can use CSS float property only for layout).

Expected Output

### The Earth!!!

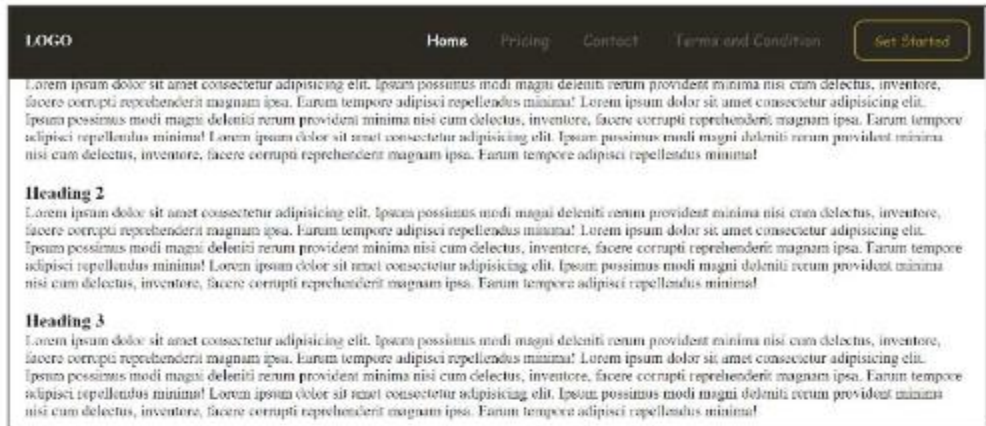
Nature is Earth's masterpiece, a symphony of life composed through millennia. Its landscapes whisper tales of ancient mountains, winding rivers, and flourishing forests, reminding us of the planet's boundless beauty. From delicate petals to towering redwoods, nature's artistry stirs awe and reconnects us to the essence of existence. Nature is Earth's masterpiece, a symphony of life composed through millennia. Its landscapes whisper tales of ancient mountains, winding rivers, and flourishing forests.





4. Create a simple header that sticks to the top of a webpage upon scrolling.

Expected Output



5. Explain the z-index, with a code example.

## Flexbox in CSS

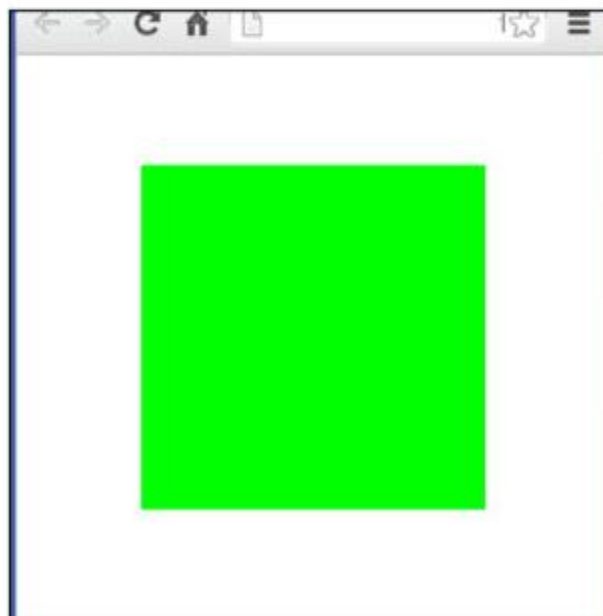
Q1. Describe the main differences between the CSS Flexbox layout model and the CSS Grid layout model. When would you choose to use one over the other?

Q2. Explain the role of the following key properties in the Flexbox layout mode

1. Justify-content
2. align-item
3. gap
4. Flex-direction
5. flex-wrap

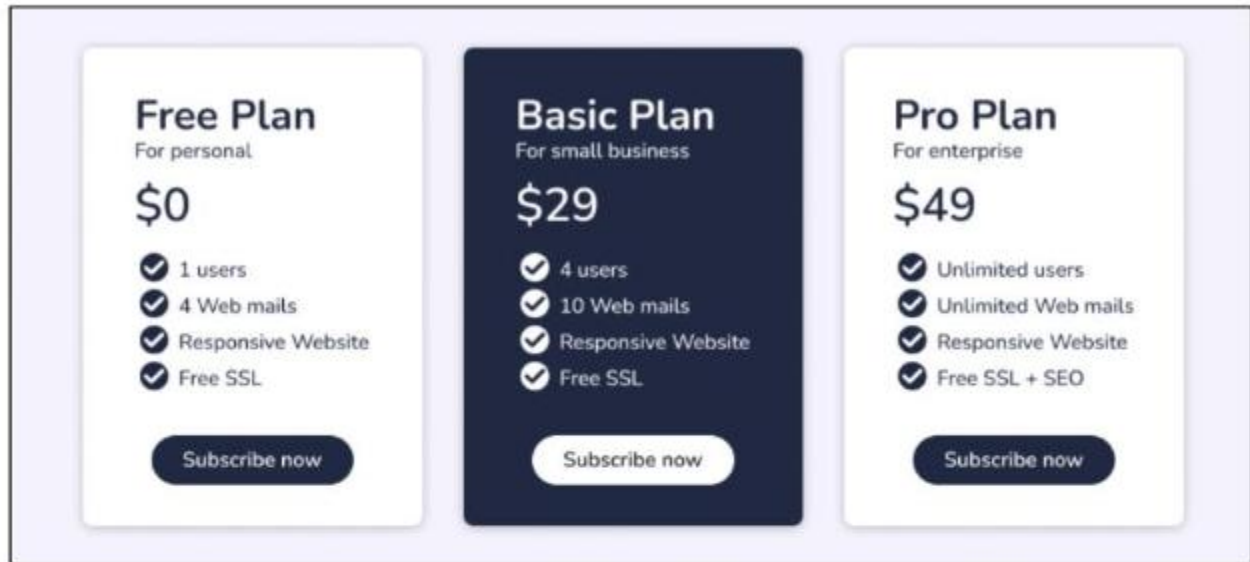
Q3. Write the code to center a div using CSS Flexbox.

Reference Image is given below.



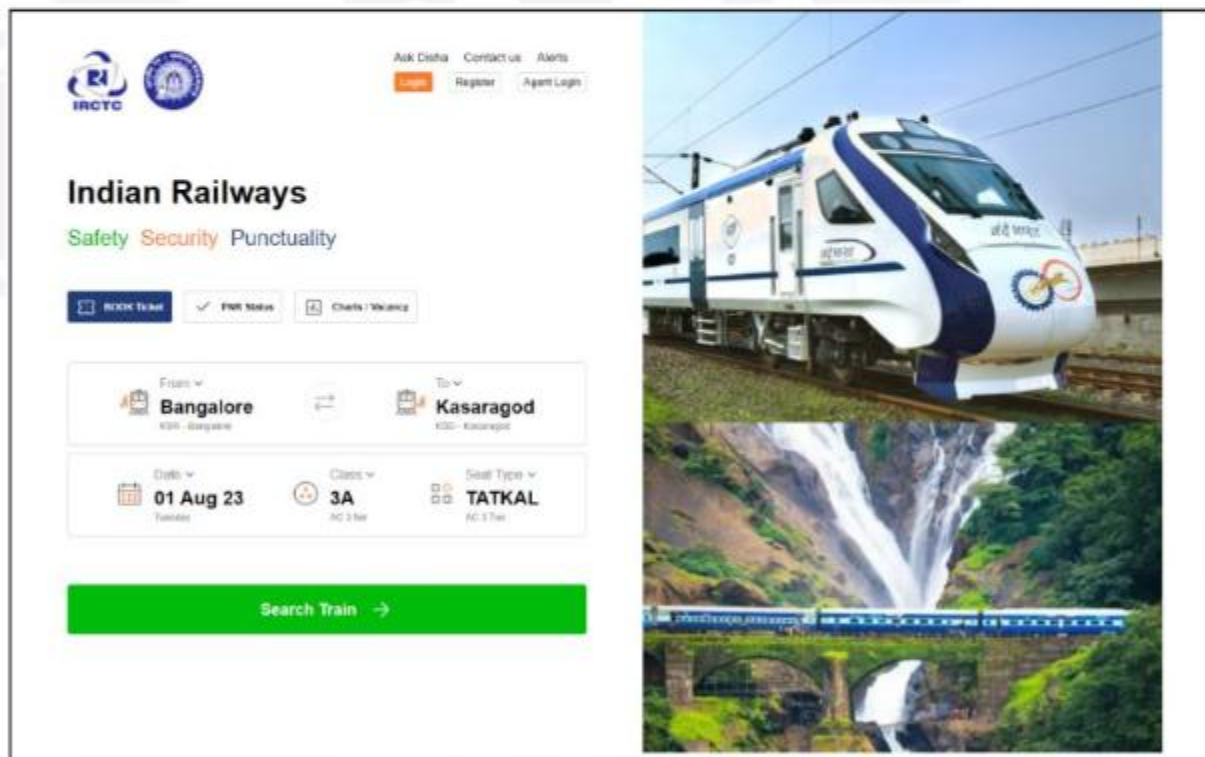


Q4. A client of yours wants to add a pricing section on their website to showcase their newly introduced premium plans. You have to build the pricing section for their business. They have provided you with the figma design for the same.[link](#)



You can refer to the Figma design and download the assets from the same.

Q5. build a clone of the IRCTC Ticket booking page.

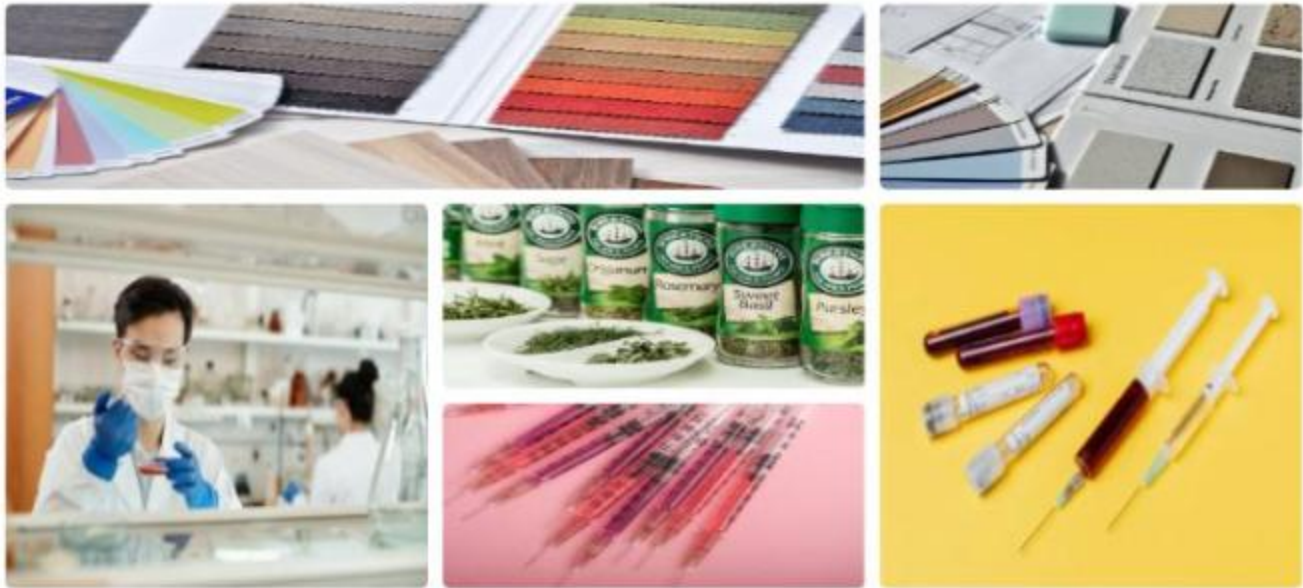


Assets can be downloaded from the figma file provided below  
Link to the figma file:[Link](#)

## Grid in CSS

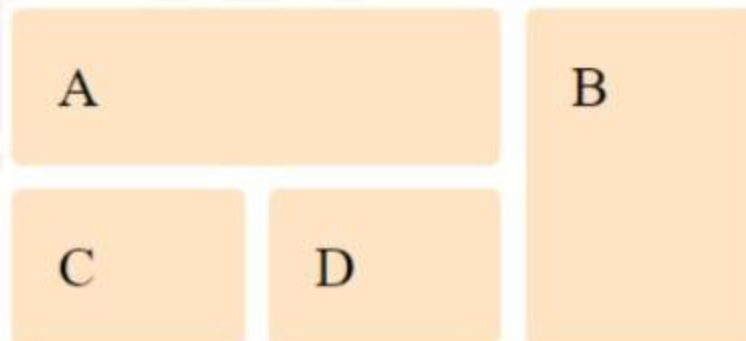
1. Create an image gallery using a CSS grid.

Expected Behaviour



2. Write code to arrange containers with texts A, B, C, and D as shown in the below image.

Expected Output



3. Explain the use of grid-auto-row and grid-auto-column using code examples.

4. Write CSS to show numbers as shown in the figure, without altering the html file.

5. Explain the difference between justify-items and justify-self using code examples.

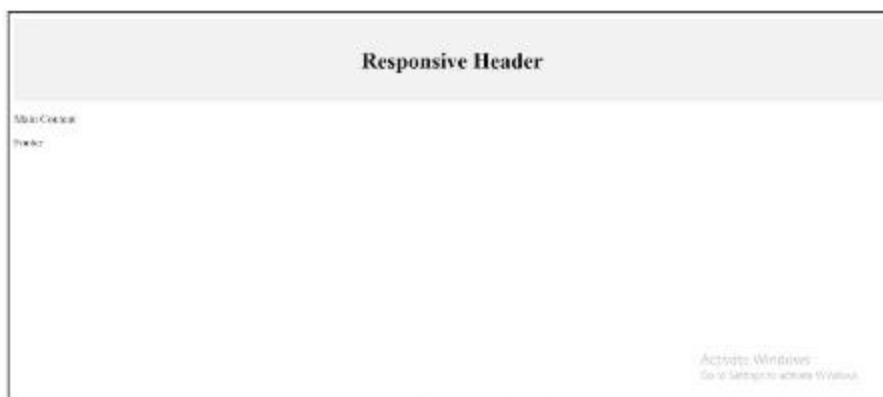


## Responsive Design

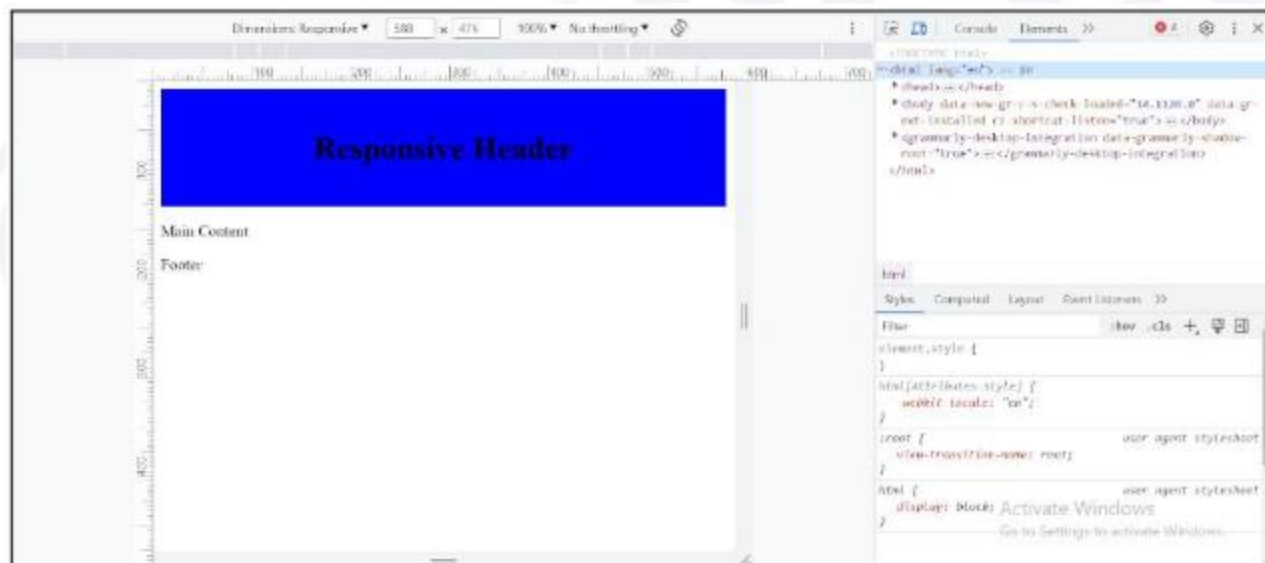
1. Create a simple webpage with a header, a main content section, and a footer. Apply a media query that changes the background color of the header to blue when the screen width is less than 600px.

### Expected Outputs

#### Normal Output



#### Responsive Output



2. Create an image gallery with three images in a row. Use media queries to adjust the layout to two images in a row for screens smaller than 800px and one image in a row for screens smaller than 500px.

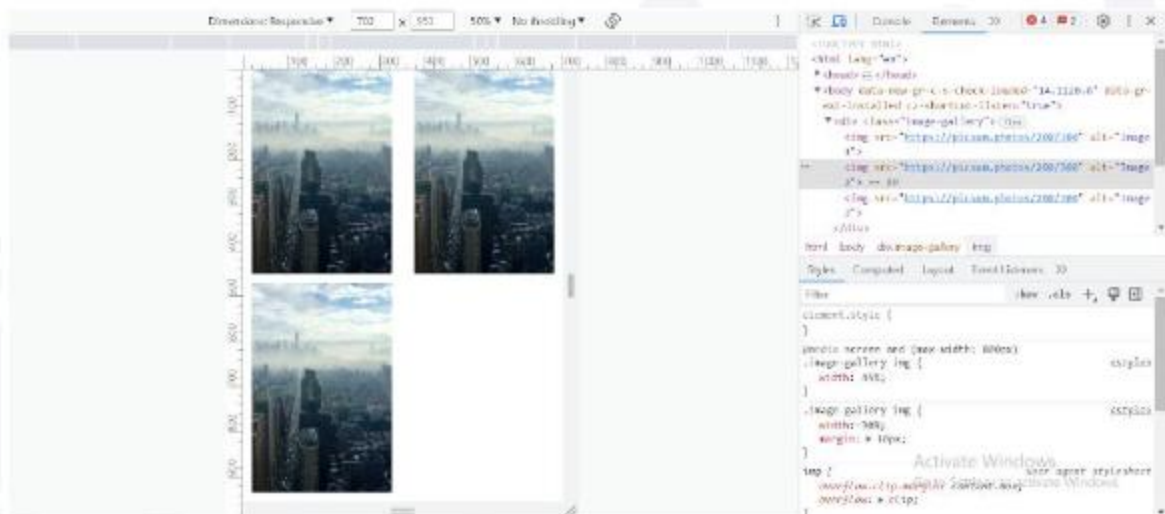
For image use this link ([Image Link](#))

### Expected Outputs

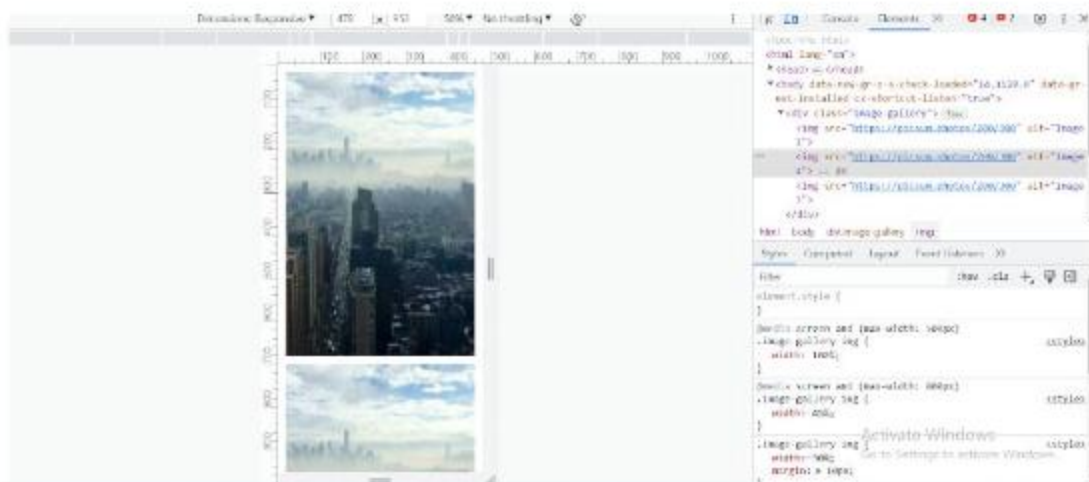
#### First Output



## Second Output



## Third Output



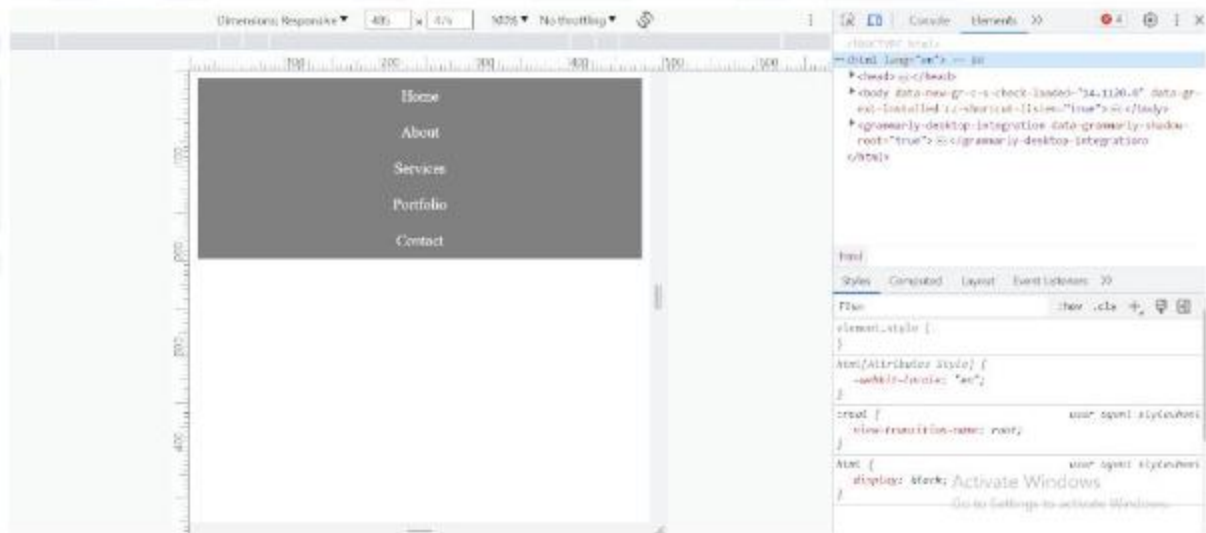


### Expected Outputs

### Normal Output

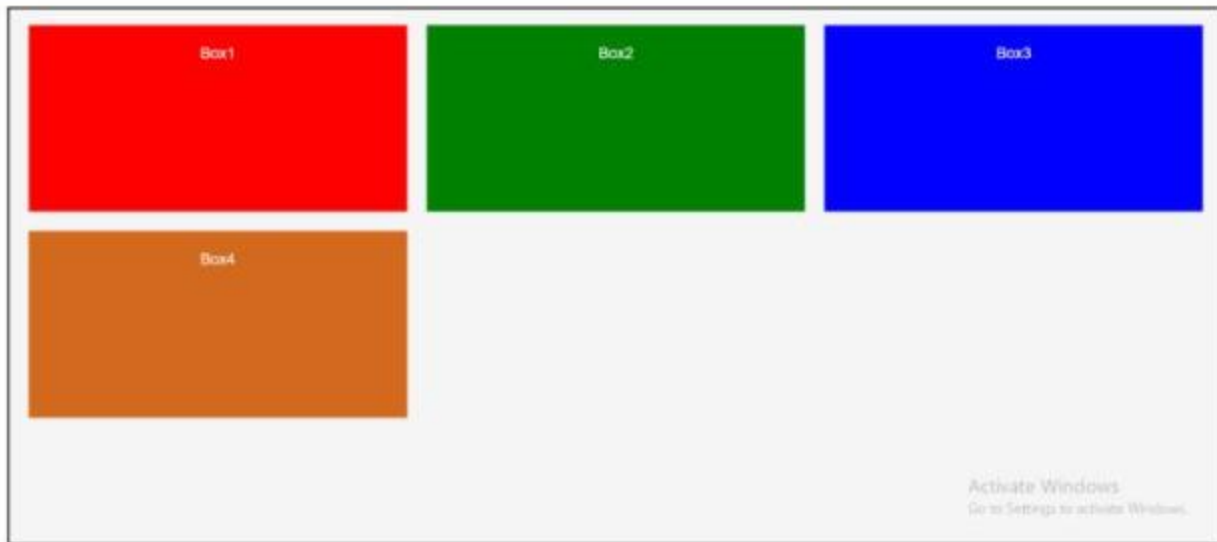


### Responsive Output

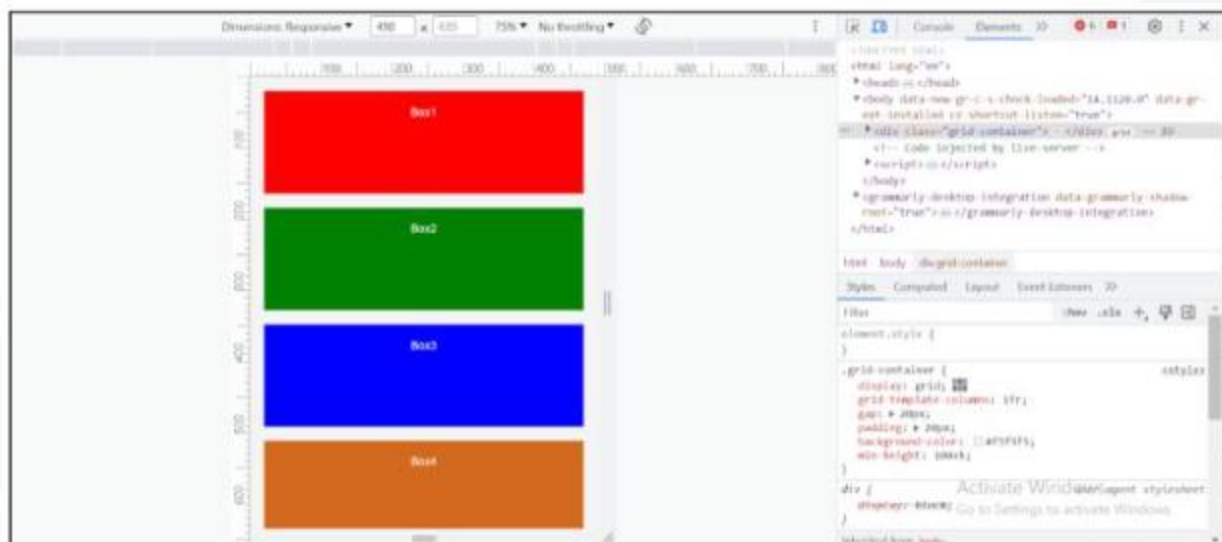


### Expected Outputs

### Normal Output



## Responsive Output



5. Create the below-given layout using the flexbox in CSS, which should adapt itself on a mobile screen as per the given below output.

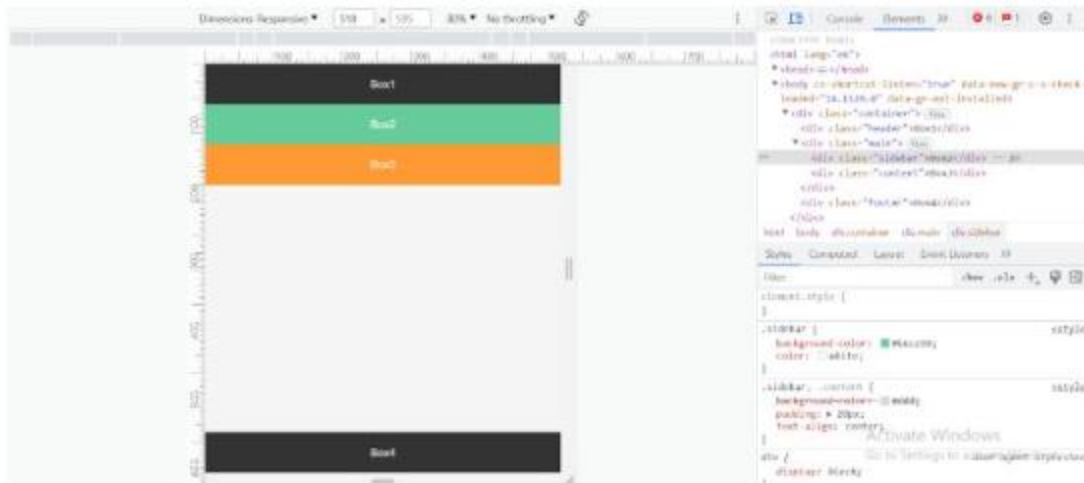
## Expected Outputs

### Normal Output





## Responsive Output



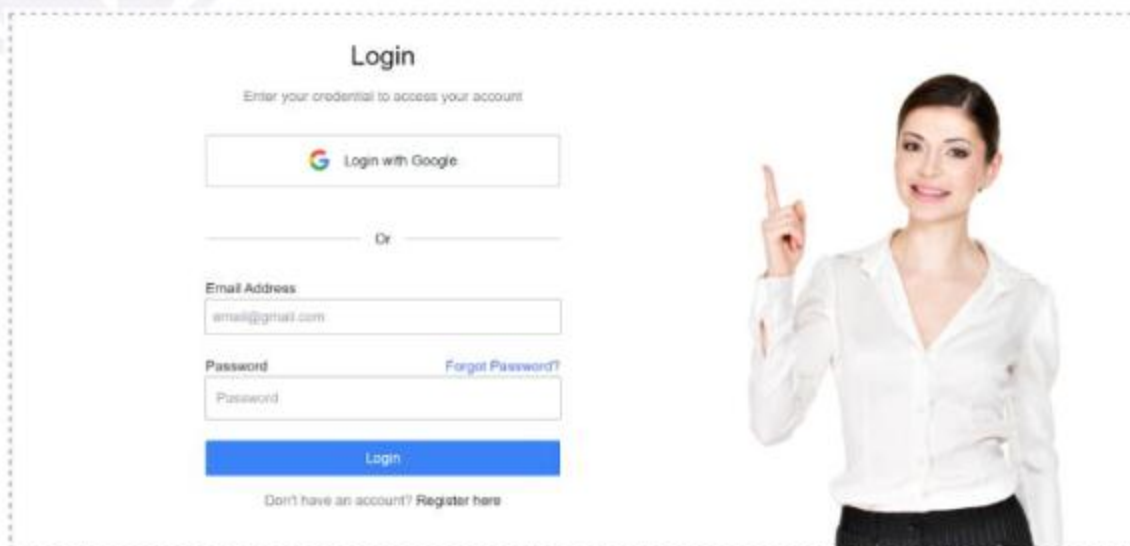
## Tailwind CSS

\* Note – all the assignments should be done using Tailwind CSS, set-up using Vite \*

1. Create a User login page interface that closely resembles the provided image using Tailwind CSS. The goal is to ensure that your implementation captures every visual detail and layout aspect depicted in the image while maintaining a full mobile responsiveness

Get the image from ([https://www.freepik.com/free-photo/young-happy-woman-with-good-idea-sign-white-shirt-full-portrait\\_10881326.htm#page=2&query=login&position=11&from\\_view=search&track=sph](https://www.freepik.com/free-photo/young-happy-woman-with-good-idea-sign-white-shirt-full-portrait_10881326.htm#page=2&query=login&position=11&from_view=search&track=sph))


Expected output –  
large screen (minimum width of 1024px)



Small screen (minimum width of 640px)

## Login

Enter your credential to access your account

 Login with Google

Or

Email Address

Password [Forgot Password?](#)

Login

[Don't have an account? Register here](#)

Extra large screen (minimum width of 1280px)

## Login

Enter your credential to access your account

 Login with Google

Or

Email Address

Password [Forgot Password?](#)

Login

[Don't have an account? Register here](#)

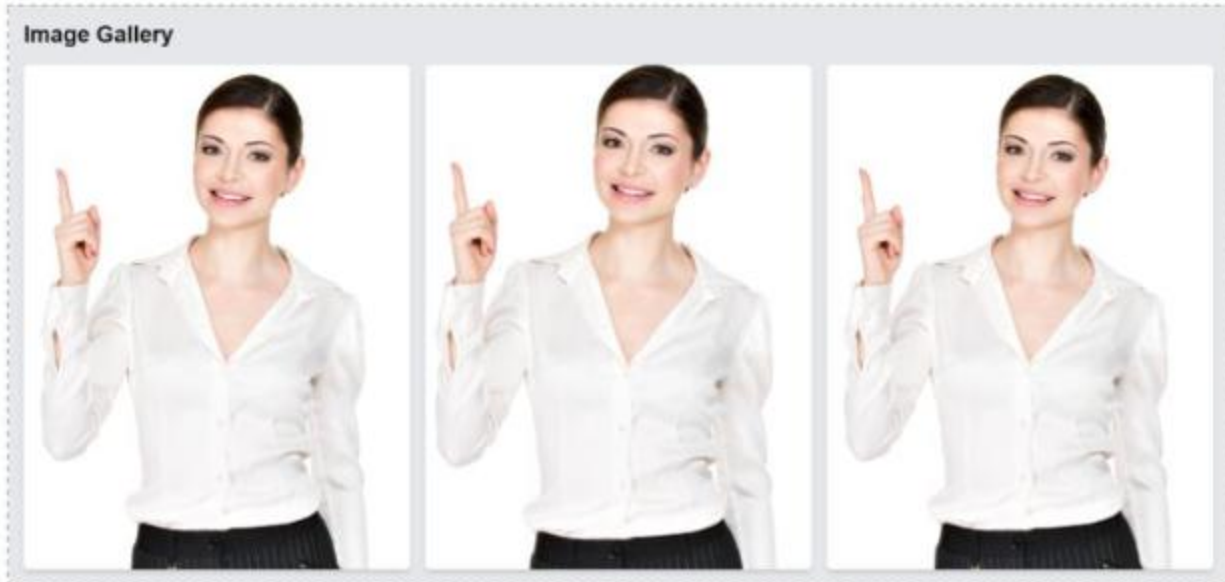


2. Create a responsive image gallery using Tailwind CSS. The gallery should display a grid of images that adjust their size and layout based on the screen size.

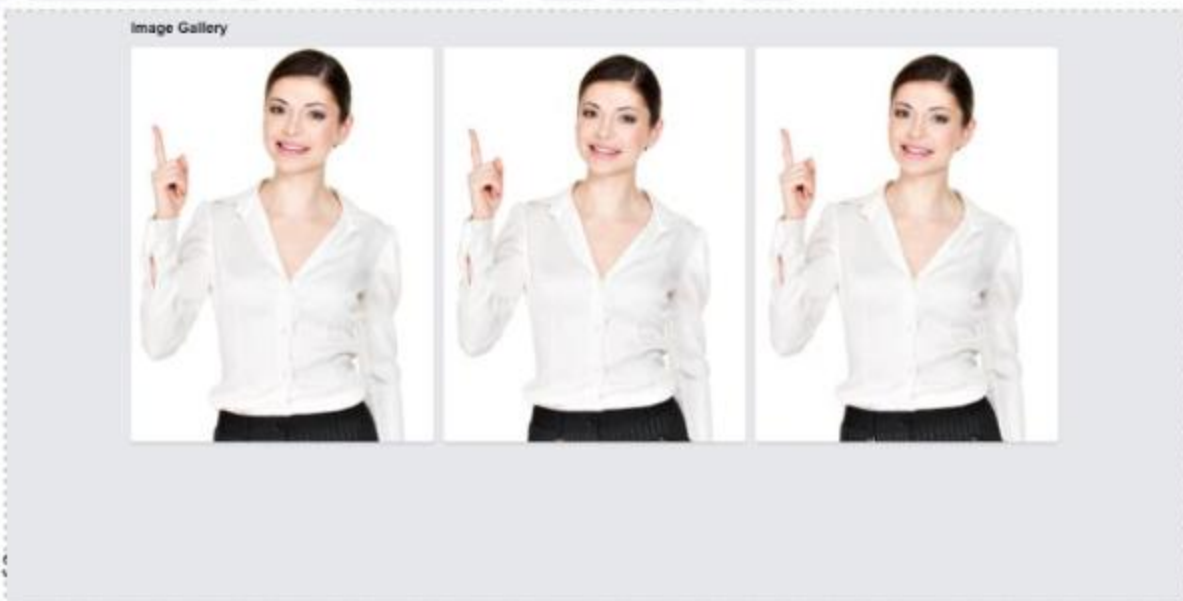
Expected output -

Expected output -

large screen (minimum width of 1024px)



Extra large screen (minimum width of 1280px)







3. Create a non-responsive dashboard using Tailwind CSS, where the dashboard should include a sidebar navigation menu and a main contain area. (Making it responsive is optional)  
Feel free to use any icons from any place, Your goal is to design a visually appealing and user-friendly dashboard interface that is not responsive.

Expected output –



**Q2. Write a program that grades students based on their marks**

- If greater than 90 then A Grade.
- If between 70 and 90 then a B grade
- If between 50 and 70 then a C grade
- Below 50 then an F grade

**Q3. What are loops, and what do we need them? Explain different types of loops with their syntax and examples.**

**Q4. Generate numbers between any 2 given numbers.**

Ex:

- const num1 = 10;
- const num2 = 25;

Output: 11, 12, 13, ..., 25

**Q5. Use the while loop to print numbers from 1 to 25 in ascending and descending order.**

## Core JavaScript – 2

**1. Create an arrow function called square that takes a number as an argument and returns its square. Use the arrow function to calculate the square of a given number and display the result.**

**2. Create a JavaScript function called generateGreeting that takes a name as an argument and returns a personalized greeting message. Use this function to greet three different people.**

**3. Write a JavaScript function called calculateTax that takes an income as an argument and returns the amount of tax to be paid. Use a closure to handle different tax rates based on income ranges. Test the function with various incomes.**

### 4. Assignment: Building a Student Management System

#### Description:

You are tasked with building a student management system using JavaScript. The system should allow you to perform various operations on a list of students, including adding, updating, deleting, and displaying student information.

#### Requirements:

Here is an initial array of students. Each student is represented as an object with the following properties: id, firstName, lastName, age, and grade.

```
const students = [  
  { id: 1, firstName: "John", lastName: "Doe", age: 20, grade: "A" },  
  { id: 2, firstName: "Jane", lastName: "Smith", age: 22, grade: "B" },  
  { id: 3, firstName: "Bob", lastName: "Johnson", age: 19, grade: "A" },  
  // Add more students as needed  
];
```



**Implement the following functions using pure JavaScript (without any external libraries or frameworks):**

- Add a Student: Create a function to add a new student to the array.
- Update Student Information: Create a function to update a student's information based on their id.
- Delete a Student: Create a function to delete a student based on their id.
- List All Students: Create a function to display a list of all students.
- Find Students by Grade: Create a function to find all students who have a specific grade.
- Calculate Average Age: Create a function to calculate the average age of all students using the array method.

**5. In the following shopping cart add, remove, and edit items**

```
=> const shoppingCart = ['Milk', 'Coffee', 'Tea', 'Honey']  
add 'Meat' in the beginning of your shopping cart if it has not been already added  
add Sugar at the end of you shopping cart if it has not been already added  
remove 'Honey' if you are allergic to honey  
modify Tea to 'Green Tea'
```

**6. The following is an array of 10 students ages:**

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

- Sort the array and find the min and max age.
- Find the median age(one middle item or two middle items divided by two)
- Find the average age(all items divided by number of items)
- Find the range of the ages(max minus min)
- Compare the value of (min - average) and (max - average), use abs() method

**7. Create a Map in JavaScript and perform the following operations:**

- Add key-value pairs to the Map.
- Check if a specific key exists.
- Retrieve the value associated with a given key.
- Iterate through all key-value pairs.

## HOF and Functional Programming

- Write a function calculate that takes two numbers and a callback function. The callback function should perform a mathematical operation (addition, subtraction, multiplication, or division) on the two numbers and return the result.
- Write a function delayedMessage that takes a message and a delay time in milliseconds. The function should log the message after the specified delay.
- Write a function printArrayElements that takes an array and logs each element using the forEach method.
- Write a function sumEvenNumbers that takes an array of numbers and returns the sum of all even numbers using the reduce method.
- Write a function calculate that takes two numbers and a callback function. The callback function should perform a mathematical operation (addition, subtraction, multiplication, or division) on the two numbers and return the result.



## Closure and Destructuring

1. Write a function counter that returns an object with two methods: `increment()` which increments a count variable initialized to 0, and `getCount()` which returns the current count value.
2. Write a function `calculateTotal` that takes an object with properties `price` and `quantity`, and returns the total cost using destructuring.
3. You are given an array of numbers named `myArray`. Create a function that takes any number of arguments and adds them to the existing array. Use the spread and rest operator.
4. Write a function `multiplyAndSum` that takes any number of arguments using the rest operator and returns the product of the first two arguments added to the sum of the rest.

## Git and GitHub

- Q1. Explain what version control is and its importance in software development
- Q2. Explain the Git Workflow, including the staging area, working directory, and repository
- Q3. Explain what `.gitignore` is and why it's important in version control
- Q4. Briefly explain what GitHub is and how it facilitates collaboration and version control also name some alternatives to GitHub.
- Q5. Describe the process of contributing to any open-source project on GitHub in a step-by-step manner.
- Q6. Deploy Tailwind projects named Youtube, slack, and Gmail clones on GitHub pages and share the deployed link of those three. Expected output - Live hosted URL Link of your deployed respective website with GitHub pages.

## JavaScript DOM

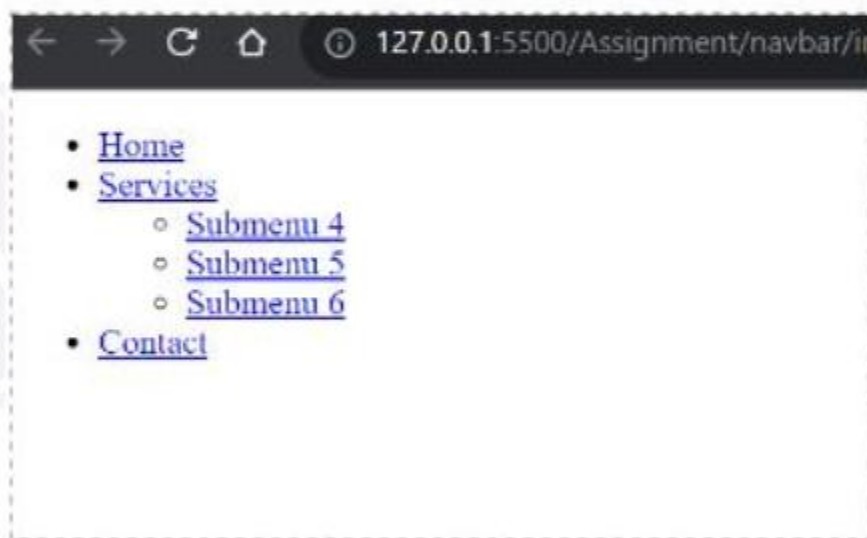
1. Explain the DOM and its role in Web development
2. Explain the concept of event delegation and provide a scenario where it is beneficial.
3. Explain the concept of Event Bubbling in the DOM
4. Explain the purpose of the `addEventListener` method in JavaScript and how it facilitates event handling in the DOM.
5. Create an HTML page with a button. Use JavaScript to display an alert when the button is clicked.
6. Create a simple image carousel using HTML and JavaScript, Design a basic HTML structure with images, and use JavaScript to implement functionality that allows users to navigate through the images.

Expected output -



7. Build a dynamic dropdown menu using HTML and JavaScript. Create an HTML structure for a navigation menu with dropdowns. Use JavaScript to toggle the visibility of dropdowns when the user hovers over menu items.

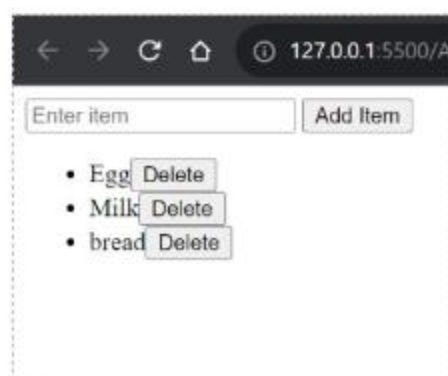
Expected output -



On hover of the menu, the sub-menu list will be displayed.

8. Create a simple dynamic shopping list with the following features
- The item should appear in the list.
  - Each item should be given a button that can be pressed to delete that item off the list.
  - The input should be emptied and focused ready for you to enter another item

Expected output -





## Walkthrough With React


- Q1. Explain the potential SEO challenges and solutions associated with Single Page Applications (SPAs).
2. Explain the key differences and use cases between React's Client-Side Rendering (CSR) and Server-Side Rendering (SSR). Provide examples of scenarios where each approach is advantageous, and discuss the challenges associated with using React in both contexts.
3. Explain the difference between npm and npx using the create-react-app code example.
4. Create a React component called "Login" without using JSX for its definition. This component should render an HTML structure that includes a form with input fields for a username and password. It should also have a button labeled "Submit."

### Expected Output

<input type="text" value="username"/>	<input type="password" value="password"/>	<input type="button" value="Submit"/>
---------------------------------------	---	---------------------------------------

5. Create a React component called "Product" using JSX. This component should render an HTML structure that includes a product image, price, name, and description. It should also have a button labeled "Buy Now."

### Expected Output



**Headphones | High Base Clear Sound**

The flagship-level battery life for the all-new OnePlus Nord Buds 2r delivers up to 38 hrs of non-stop music on a single charge.

\$12

## Router & hooks

1. Discuss the concept of hooks in React. Explain how they differ from class component lifecycle methods. Also, provide an example
2. Explain and discuss its syntax, purpose, and how it manages state in functional components for the following hooks
  - useState
  - useEffect
3. Create a global state management system using the Context API and useContext hook to build a theme toggle React mini project. Follow these steps to implement a light/dark theme toggle feature.



## Steps to Complete the Assignment

### 1. Create a Theme Context

- Define a theme context in a new file.

### 2. Create a Theme Provider

- Define the ThemeProvider component to provide the theme and toggle function using the useState hook.

### 3. Use the ThemeProvider in Your App

- Wrap your application with the ThemeProvider in the main application file.

### 4. Create a Themed Component

- Create a component that consumes the theme using the useContext hook.

## Browser output -

### Before clicking on the Theme toggle



### After clicking on the theme toggler



## 4. Set up a basic React Router v6 application with multiple pages and navigation.

### Task:

- Create a new React application using the Create React App.
- Install react-router-dom.
- Set up the following routes:
  - Home (/)
  - About (/about)
  - Contact (/contact)
- Create corresponding components for each route.
- Use the Link component to navigate between the pages.

# Form Handling & State Management

## Q1. Create a simple form with controlled components in React.

### 1. Task:

- Create a form component (LoginForm) with fields for username and password.
- Implement controlled components for each input field.
- Display validation errors if either field is empty upon form submission.
- Log the form data to the console upon successful submission.

### 2. Deliverables:

- A LoginForm component that uses controlled components for username and password.
- Validation logic to display errors if fields are empty.
- Console log of form data upon successful submission.

## Q2. Implement form validation using React state and conditional rendering.

### 1. Task:

- Create a form component (RegistrationForm) with fields for name, email, and password.
- Implement state to manage form inputs and validation errors.
- Validate name (required, minimum length), email (required, valid format), and password (required, minimum length).
- Display error messages next to each field if validation fails.
- Disable the submit button until all fields are valid.
- Alert the user with a success message upon valid form submission.

### 2. Deliverables:

- A RegistrationForm component with controlled inputs and validation state.
- Validation rules for name, email, and password.
- Error messages are displayed conditionally based on validation status.
- Disabled/enabled the submit button based on form validity.
- Alert message upon successful form submission.

## 3. Set up Redux store, actions, and reducers for basic state management.

### 1. Task:

- Initialize a new React application with Redux using Create React App.
- Set up Redux store, actions, and reducers.
- Create a feature for managing a counter:
  - Implement actions (increment, decrement) and corresponding reducers to update the counter state.
  - Display the counter value in a React component (Counter) connected to the Redux store.
  - Include buttons to increment and decrement the counter.

### 2. Deliverables:

- A Redux store configured with actions and reducers for managing a counter.
- A Counter component displaying the counter value and buttons to update it.
- Proper integration of Redux with React components using connect or useSelector and useDispatch hooks.

# MongoDB

## 1. Understanding Schemas in MongoDB

### Task:

- Explain what a schema is in the context of MongoDB.
- Discuss how schemas are defined and enforced in MongoDB using Mongoose (or any other ODM).
- Create a simple schema for a blog post, including fields for title, author, content, tags, and createdAt.

2. Model a simple e-commerce application with collections for users, products, and orders, using both embedded and referenced data models.

3. Creating and Dropping a Database in MongoDB

### Task:

1. Explain how to create and drop a database in MongoDB.
2. Write a script to create a new database named shopDB.
3. Write a script to drop the shopDB database.

## 4. Creating and Dropping a Collection in MongoDB

### Task:

1. Explain how to create and drop a collection in MongoDB.
2. Write a script to create a collection named products in the shopDB database.
3. Write a script to drop the products collection from the shopDB database.

## 5. Create a sample document for a product collection, using at least five different data types.



## Nodejs

### 1. Setting Up a Basic Node.js Server

#### Task:

1. Create a basic Node.js server that listens on port 3000.
2. Respond with "Hello, World!" for requests to the root URL (/).
3. Use the built-in http module to create the server.

### 2. Working with Modules in Node.js

#### Task:

1. Explain what modules are in Node.js and why they are important.
2. Create a custom module that exports a function to calculate the area of a rectangle.
3. Import and use this custom module in a separate script to calculate and log the area of a rectangle with given dimensions.

### 3. Handling Asynchronous Operations in Node.js

#### Task:

1. Explain the difference between synchronous and asynchronous operations in Node.js.
2. Create a script that reads a file asynchronously and logs its contents.
3. Use the built-in fs module to handle the file reading.

### 4. Interacting with the File System

#### Task:

1. Create a script that writes a string to a new file.
2. Append additional text to the same file.
3. Read and log the contents of the file to the console.

### 5. Explain what Node.js is and its primary use cases.

## Express

### 1. Explain Express.js and discuss why it is commonly used in web development.

### 2. Implement a basic Express.js server with multiple routes.

#### Details:

- Set up an Express.js server that listens on a specific port (e.g., 3000).
- **Define routes for:**
  - **GET /:** Display a "Hello World!" message.
  - **GET /about:** Display information about your application or yourself.
  - **GET /api/users:** Return mock JSON data representing a list of users.
- Ensure the server responds correctly to each route and handles potential errors.



**Expected Output:**

- Working code that initializes an Express.js server and defines the specified routes.
- Demonstration of using Express.js routing methods (app.get, app.post, etc.) effectively.

**3. Implement protected routes using middleware in Express.js.****Details:**

- **Create an Express.js server with routes:**
  - **GET /:** Display a "Welcome to the main page!" message.
  - **GET /dashboard:** Display a dashboard page accessible only if authenticated.
- Implement middleware (checkAuth) to verify authentication status (e.g., using a boolean flag).
- Ensure GET /dashboard route is only accessible when authenticated; otherwise, redirect to GET /.

**Expected Output:**

- Code demonstrating middleware usage (checkAuth) to protect routes.
- Clear demonstration of route protection mechanism and error handling.

## Express App – 1

**1. Creating a User Profile (POST)****Task:**

- Implement an endpoint to create a new user profile.
- The profile should include fields: id, name, email, and age.
- Use the POST /profiles endpoint to create a profile.

**Details:**

- Accept profile data in the request body.
- Store the profile data in an in-memory array or object for simplicity.
- Return the created profile as a response.

**Deliverables:**

- Code that defines the POST /profiles route.
- Example request:

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "age": 30
}
```

## 2. Fetching User Profiles (GET)

### Task:

1. Implement an endpoint to fetch all user profiles.
2. Implement an endpoint to fetch a specific user profile by id.

### Details:

- Use the GET /profiles endpoint to return all profiles.
- Use the GET /profiles/:id endpoint to return a profile by its id.

### Deliverables:

- Code that defines the GET /profiles and GET /profiles/:id routes.
- Example response for GET /profiles:

```
[
  {
    "id": 1,
    "name": "John Doe",
    "email": "john@example.com",
    "age": 30
  },
  ...
]
```

- Example response for GET /profiles/:id:

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com",
  "age": 30
}
```

## 3. Updating a User Profile (PUT)

### Task:

1. Implement an endpoint to update an existing user profile by id.
2. Use the PUT /profiles/:id endpoint to update profile details.

### Details:

- Accept updated profile data in the request body.
- Update the corresponding profile in the in-memory data store.
- Return the updated profile as a response.

### Deliverables:

- Code that defines the PUT /profiles/:id route.
- Example request:

```
{
  "name": "Jane Doe",
  "email": "jane@example.com",
  "age": 28
}
```

## 4. Deleting a User Profile (DELETE)

### Task:

1. Implement an endpoint to delete a user profile by id.
2. Use the DELETE /profiles/:id endpoint to remove a profile.

### Details:

- Remove the profile with the specified id from the in-memory data store.
- Return a success message or status.

### Deliverables:

- Code that defines the DELETE /profiles/:id route.
- Example response:

```
{
  "message": "Profile deleted successfully"
}
```