HTML QUESTIONS

Q.1 <!DOCTYPE html> is it a tag of html? If not, what is it and why do we use it?

Ans:<!DOCTYPE html> is not an HTML tag.
Doctype stands for **Document Type Declaration**. It informs the web browser about the type and version of HTML used in building the web document. This helps the browser to handle and load it properly.While the HTML syntax for this statement is somewhat simple, you must note Each version of HTML has its own rules.

Q.2 Explain Semantic tags in html? And why do we need it?

Ans:.Semantic tags define the purpose of the HTML element. It also describes the type of content that the element should contain.
For example, <header>, <footer>, <article>, , <li>, <section>, , <main>, <address>, <time> are considered semantic elements as they clearly describe their purpose and the type of content they should enclose.

Q.3 Differentiate between HTML Tags and Elements?

Ans: The **HTML tag** is just an opening or closing entity of a HTML element.
For Example: and are called Html tags.
An **Html Element** consists of an opening and closing HTML tag with the content inserted in between the HTML tag.
For Example: <h1>hello world</h1> are called Html elements.

Q.4 Build Your Resume using HTML only.

Ans:
Github Link: https://github.com/gerard0lucas/my-resume-using-Html
Deployed Link: https://shiny-madeleine-291cc9.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.5 Write Html code so that it looks like the given image Link

Ans:
Github Link: https://github.com/gerard0lucas/placement-assigngent-Q5-Html-gerard
Deployed Link: https://luxury-mandazi-02d723.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.6 What are some of the advantages of HsTML5 over its previous versions?

Ans:HTML 5 is the latest version of HTML which is the core markup language of the World Wide Web that can be written in both HTML (HTML 5) and XML (XHTML 5) language.
HTML 5 allows usage of new structural elements and attributes, replacing some of the familiar elements. Moreover it's providing some fresh functionality like audio and video elements which will offer an easy way to semantically structure the page.

Q.7 Create a simple Music player using html only.

Ans: This is a simple audio player with audio tag.
Github Link: https://github.com/gerard0lucas/audio-player
Deployed Link: https://beautiful-salmiakki-8f068b.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.8 What is the difference between <figure> tag and <audio> tag?

Ans: The **<img>** tag is used to display an image such as a photograph or illustration on the web page.
The **<figure>** tag is used as a container to hold an image, graph or other illustration on the web page.

Q.9 What's the difference between html tag and attribute and give example of some global attributes?

Ans: The **HTML tag** is just an opening or closing entity of a HTML element.
For Example: and are called Html tags.
Attributes provide additional information about HTML elements.
For Example:Google
The **<a>** tag defines a hyperlink. The **href** attribute specifies the URL to be redirected.

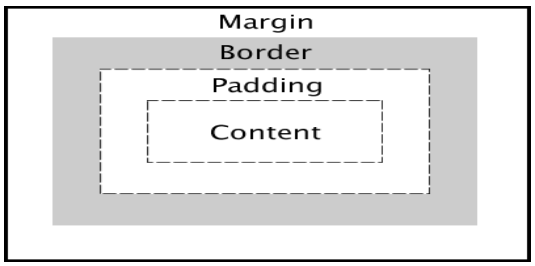
Q.10 build Table which looks like the given image Link

Ans:
Github Link: https://github.com/gerard0lucas/Table-using-html
Deployed Link: https://prismatic-queijadas-dac3fb.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

CSS QUESTIONS

Q.1 What's Box Model in CSS & Which CSS Properties are part of it ?

Ans: Every element that can be displayed is considered as box. In CSS, the term 'box model' is used to represent margin, border and padding of the HTML element. The image below illustrates the box model:



The innermost box holds the HTML element's content. Between the box contents and the element border lies the padding. Padding size can be set on all four sides of the content. It inherits its color from the background-color property of the content. The margin lies outside the border and gets its color from the parent element.

Q.2 What are the Different Types of Selectors in CSS & what are the advantages of them?

Ans: A selector in CSS is a part of the CSS ruleset, that is basically used to select the element you want to style. CSS selectors select HTML elements according to their id, class, type, attribute, etc.

1. Universal selector

The universal selector selects all the HTML elements on the page. All the elements are selected on the page using (*)

2. Individual selector

The individual selectors select the HTML elements by name.

3. Id selector

The id selector selects the element based on the id attribute. The id of an element is unique within the page, so the id selector is used to select the unique element. It is written with the hash character (#), followed by the id of an element.

4. class selector

The class selector selects the element based on the class attribute. It is written with a period character (.), followed by the class name.

5. Combined selector

The combined selector is used to select all the elements with the same style definition. Commas(,) are used to separate the elements in the grouping. This is use to select multiple elements at same time

6. Inside an element

If you want to specify that only one specific HTML element should be affected then you should use the element name with class selector or id selector or child or sibling element.

.

Q.3 What is VW/VH & How is it different from PX?

Ans: VW/VH is equal to 1% of the viewport's height and viewport's width, with the viewport being the browser window. 1px is one physical pixel on the device's screen. Percent is the relative size compared to its parent element. For example, a div with 50% width inside another div will take up half of the outer div's width. VW/VH and percent are both relative units, so they scale based on some factor of the device it is viewed on. Pixels are absolute, so a 100x100 pixel image will be larger on a low-resolution screen, but very small on a screen with high pixel density.

Q.4 What's the difference between Inline, Inline Block and block ?

Ans:

1.Block-level elements

Take full width (100% width) by default, Each gets displayed in a new line, Width & height properties can be set and Can contain other block or inline elements.

2.Inline elements

Take only as much space as they need, displayed side by side, An inline element cannot contain a block-level element but Can be a parent of other inline elements.

3.Inline-block elements

The CSS display: inline-block is a combination of both inline and block-level features. The main difference is that inline-block responds to width and height properties. This feature makes the CSS display: inline-block more suitable for creating layouts. One of the more popular ways of using inline-block elements is creating horizontal navigation menus.

Q.5 How is Border-box different from Content Box?

Ans: The main difference between border-box and content-box is how they calculate the total width and height of an element:

Content-box: This is the default value. The total width and height of an element are calculated based on the width and height of its content, without including the padding or border. This means that any padding or border you add to the element will increase its overall size.

Border-box: The total width and height of an element are calculated based on the width and height of the content, padding, and border combined. This means that any padding or border you add to the element will be included in its overall size. The content area is then calculated as the remaining space within the element.

Q.6 What’s z-index and How does it Function ?

Ans:z-index is the CSS property that controls the stacking order of overlapping elements on a page. An element with a higher z-index value will appear in front of an element with a lower z-index value.
The functioning of x-index is like it sets the order of elements along the z-axis. If the x-axis goes left-to-right and the y-axis goes top-to-bottom, the z-axis adds the dimension of “toward” and “away from” the user. Elements with a higher z-index value appear closer to the user, and elements with a lower value look farther away.

Q.6 What’s Grid & Flex and difference between them?

Ans: Flexbox and CSS Grid are two layout modules in CSS that are used to create responsive, flexible layouts for web pages. Both have their own advantages and can be useful in different situations.

Flexbox: The CSS Flexbox offers a one-dimensional layout. It is helpful in allocating and aligning the space among items in a container (made of grids). It works with all kinds of display devices and screen sizes. To get started you have to define a container element as a grid with **display: flex;**

Example:

```
<html>
<head>
<style>
.flex-container {
  display: flex;
  justify-content: space-around;
  background-color: Blue;
}

.flex-container div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
}
</style>
</head>
<body>

<h1>Create a Flex Container</h1>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
</body>
</html>
```

Grid: -CSS Grid Layout, is a two-dimensional grid-based layout system with rows and columns, making it easier to design web pages without having to use floats and positioning. Like tables, grid layout allow us to align elements into columns and rows.
To get started you must define a container element as a grid with **display: grid;** and set the column and row sizes with grid-template-columns and grid-template-rows, and then place its child elements into the grid with grid-column and grid-row.

Example:

```
<html>
<head>
<style>
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
</style>
</head>
<body>
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div class="grid-item">9</div>
</div>
</body>
</html>
```

Q.7 Difference between absolute and relative and sticky and fixed position explain with an example.

Ans:

1. Absolute Positioning

In Absolute Positioning, the element ignores the normal document flow, but rather than being positioned according to the viewport, it’s positioned relative to the nearest non-static ancestor. However, like the fixed value, there's no space created for it in the page layout.
If the positioned element can’t find any non-static ancestor, it’s positioned relative to the web page.

Example:

```
<html>
<head>
<style>
body{
background-color: rgb(245, 116, 116);
}
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid white;
}
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid black;
}
</style>
</head>
<body>
<h2>position: absolute;</h2>
<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>
</body>
</html>
```

Output:[LINK]

2. Relative Positioning

In Relative Positioning, the element follows the render flow but will be shifted relative to its initial position. To determine the amount of offset, set values for the top, right, bottom, and left properties. The other elements around the one in relative position aren't affected, but there will be extra space where the element would have been.

Example:

```
<html>
<head>
<style>
body{
background-color: skyblue;
}
div.relative {
  position: relative;
  left: 70px;
  border: 3px solid #73AD21;
  background-color: black;
  color: white;
}
</style>
```

```
</head>
<body>
<h2>position: relative;</h2>
<div class="relative">
This div element has position: relative;
</div>
</body>
</html>
```

Output: LINK

3. Sticky Positioning

The position sticky is a mix of position: relative and position: fixed. It acts like a relatively positioned element until a certain scroll point and then it acts like a fixed element and allows you to position an element based on scroll position. Set an element as sticky on the top when a user scrolls down.

Example:

```
<html>
  <head>
    <style>
      div.sticky {
        position: -webkit-sticky;
        position: sticky;
        top: 10px;
        padding: 10px;
        background-color: rgb(80, 252, 13);
        border: 1px solid blue;
      }
    </style>
  </head>
  <body>
    <h1>CSS position:sticky Example</h1>
<div class = "sticky">After you scroll its offset point, this element will stick 10px from top of viewport.
</div>
    <div style="padding-bottom:2000px">
      <h2>Lets scroll up and down</h2>
    </div>
  </body>
</html>
```

Output: LINK

4. Fixed Positioning

The fixed position always sticks to a specific location and it can’t be moved any side of the page. Even if we minimise or maximise the page, its position is fixed only. The element disregards the normal render flow. Instead, the element is positioned relative to the viewport . It doesn’t move, even if the page is scrolled. There's no space left where it would have been located in the page layout. Use the top, right, bottom, and left values to set the element's final position.

Example:

```
<html>
<head>
<style>
div.fixed {
  position: fixed;
  top:0px;
  width:100%;
  height:130px;
  background-color: orange;
  color: white;
 text-align: center;}
</style>
</head>
<body>
<div class="fixed">I AM FIXED</div>
</body>
</html>
```

Output: LINK

Q.8 Build Periodic Table as shown in the below image.

Ans:

Github Link: https://github.com/gerard0lucas/perodic-table

Deployed Link: https://benevolent-starship-eb6934.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.9 Build given layout using grid or flex see below image for reference .

Ans:

Github Link: https://github.com/gerard0lucas/simple-layout

Deployed Link:https://endearing-lollipop-3e03f7.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.10 Build Responsive Layout both desktop and mobile and Tablet, see below image for reference ?

Ans:

Github Link: https://github.com/gerard0lucas/simplelayout-2

Deployed Link: https://harmonious-granita-46081d.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.11 Build Complete Homepage of Ineuron (Link) with responsiveness.

Ans:

Github Link: https://github.com/gerard0lucas/ineuron-home-page

Deployed Link: https://spectacular-cocada-4db4fc.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.12 What are Pseudo class in CSS & How its different From Pseudo Elements?

Ans:

Pseudo classes lets you style an element based on its state, Like (:checked, :valid, :disabled). This also implicates that they refer to an existing element: something that is already in the DOM available to style.

Pseudo elements let you style things that are not actually elements. They can be parts of existing elements (::first-letter, ::first-line), including parts that exist temporarily (::selection). Generated content also falls within the pseudo elements bracket (::before, ::after).

JavaScript Questions

Q.1 What is Hoisting in Javascript ?

Ans: Hoisting is JavaScript's default process of moving the variables and functions to the top of their scope before the code is executed. It allows us to access the variables and functions anywhere in their scope, no matter where they are declared.
Hoisting happens because JavaScript wants to find all the variables and functions before running the code to reserve memory for them.

Example:

Generally, we declare variables like this:

```
var num1; // declaration
num1 = 100; // initialization
console.log(num1); // prints 100
```

In JS, we can do the initialization before the declaration.

```
num1 = 100; // initialization
var num1; // declaration
console.log(num1); // prints 100
```

Function Hoisting:

>Similar to variables, we can call a function before its declaration.

```
printMessage();
function printMessage() {
 console.log("SemicolonSpace");
}
```

>If there is no hoisting, we have to write like this always:

```
function printMessage() {
   console.log("SemicolonSpace");
  }
  printMessage();// The function can be called only after declaration
```

Q.2 What are different higher order functions in JS? What is the difference between .map() and .forEach() ?

Ans: Higher-order functions are functions that accept another function as an argument, return another function as a result, or both.
One of the core benefits of higher-order functions is that they allow us to customise the way we call our functions.
map() - creates a new array with the results of calling a provided function on every element in the array.
forEach() - executes a provided function once for each Element in the array.

Q.3 What is the difference between .call() .apply() and .bind()? explain with an example.

Ans:

Bind( ): The bind method creates a new function and sets the **this** keyword to the specified object.

Example:

```
const john = {
 name: 'John',
 age: 24,
};
//Let's add a greeting function:
function greeting() {
 console.log(`Hi, I am ${this.name} and I am ${this.age} years old`);
}
const greetingJohn = greeting.bind(john);
// Hi, I am John and I am 24 years old
greetingJohn();
```

Call ( ): The call method sets the **this** inside the function and immediately executes that function.

Example:

```
function greeting() {
console.log(`Hi, I am ${this.name} and I am ${this.age} years old`);
}
const john = {
 name: 'John',
 age: 24,
};
const jane = {
 name: 'Jane',
 age: 22,
};
// Hi, I am John and I am 24 years old
greeting.call(john);
// Hi, I am Jane and I am 22 years old
greeting.call(jane);
```

Apply ( ): The **apply()** method is similar to **call()**. The difference is that the **apply()** method accepts an array of arguments instead of comma separated values.

Example:

```
function greet(greeting, lang) {
 console.log(lang);
 console.log(`${greeting}, I am ${this.name} and I am ${this.age} years old`);
}
const john = {
 name: 'John',
 age: 24,
};
const jane = {
 name: 'Jane',
```

```
 age: 22,
};
// Hi, I am John and I am 24 years old
greet.apply(john, ['Hi', 'en']);
// Hi, I am Jane and I am 22 years old
greet.apply(jane, ['Hola', 'es']);
```

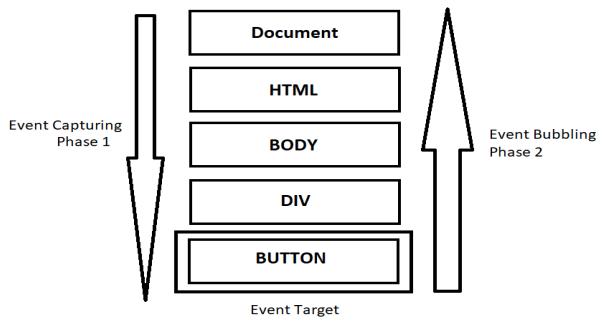
Q.4 Explain Event bubbling and Event Capturing in JavaScript with suitable examples.

Ans:

Event capturing is the scenario in which the events will propagate, starting from the wrapper elements down to the target element that initiated the event cycle.

Event bubbling is pretty simple to understand if you know event capturing. It is the exact opposite of event capturing.Event bubbling will start from a child element and propagate up the DOM tree until the topmost ancestor's event is handled

Event bubbling and Event Capturing Diagram for understanding.



Q.5 What is function currying with an example?

Ans: A currying function accepts one input at a time and outputs a new function that anticipates the subsequent argument. It involves changing a function from being callable as **f(a, b, c)** to be callable as **f(a)(b) (c)**.

Example:

```
//Non Curried version
 const add = (a, b, c)=>{
 return a+ b + c
 }
 console.log(add(2, 3, 5)) // output: 10
//Curried version
const add curry =(a) => {
 return (b)=>{
 return (c)=>{
 return a+b+c } } }
 console.log(addCurry(2)(3)(5)) // output: 10
```

Q.7 What are promises? What are the different states of a promise? Support your answer with an example where you need to create your own promise.

Ans: Promises are used to handle asynchronous operations in JavaScript.
The promise is a constructor which takes only one argument which is a callback function.The callback function takes two arguments, **resolve** and **reject**.
The operations inside the promise is like if everything went well then call resolve.
If desired operations do not go well then call reject.
JavaScript promises have three states: **pending**, **resolved**, and **rejected**.
pending: initial state, neither fulfilled nor rejected.
resolved: meaning that the operation was completed successfully.
rejected: meaning that the operation failed.

Example:

```
function checkadult (a) {
  const customPromise = new Promise((resolve, reject) => {
    const age = a ;
         if(age => 18){
      resolve("Let's go! You are adult")
    } else {
      reject(new Error('Oops!.. You re not old enough'))
    }
  })
       return customPromise
}
// consuming the promise
   checkadult(15).then(data => {
   console.log(data)
})
.catch(err => {
  console.log(err)
})
```

Q.8 What is ‘this’ keyword in JavaScript? explain with an example.

Ans: In javascript **this** keyword refers to the current object, which can be the global object, a function, or an object created by a constructor.The value of **this** changes based on the context in which it is used. There are four primary contexts for this in JavaScript.
Global Context: When this is used in the global context (outside of any function), it refers to the window object in the browser.
Example:

```
function myFunction() {
 console.log(this);
}

myFunction(); // logs window object
```

Function Context: When this is used inside a function, it refers to the object that the function is a method of or the global object if the function is not a method of any object.

Example:

```
const myObj = {
 name: 'John',
 greet: function() {
   console.log(`Hello, my name is ${this.name}`);
 }
};

myObj.greet(); // logs 'Hello, my name is John'
```

Object Context: When `this` is used inside an object method, it refers to the object itself.

Example:

```
function greet() {
 console.log(`Hello, my name is ${this.name}`);
}
const myObj = {
 name: 'John',
};

greet.call(myObj); // logs 'Hello, my name is John'
```

Constructor Context: When `this` is used inside a constructor function, it refers to the object that is being created.

Example:

```
function Car(make, model) {
 this.make = make;
 this.model = model;
}

var myCar = new Car('Honda', 'Civic');
```

Q.9 Explain event loop Call Stack Callback queue and Micro Task queue in Your Words.

Ans:

Call Stack is a data structure that keeps track of the execution of JavaScript code.

Microtask Queue is synchronous blocks of code with exclusive access to the Call Stack while executing. Additionally, they are stored in their own FIFO (First In, First Out) data structure.

Event Loop is a loop that keeps running and checks if the Call Stack is empty. It processes Tasks and Microtasks, by placing them in the Call Stack one at a time and also controls the rendering process.

Callback queue is a queue of tasks that are executed after the current task. The callback queue is handled by the JavaScript engine after it has executed all tasks in the microtask queue

Q.10 Explain Debouncing and Create a project where you are using Debouncing.

Ans: Debouncing in Javascript is a technique for improving browser speed during lengthy calculations. If this method is called frequently, it may damage the performance of our web app. Debouncing is a programming technique for reducing the frequency with which a time-consuming process fires. In other words, it restricts the frequency with which functions are called.

Example:

Github Link: https://github.com/gerard0lucas/debounce

Deployed Link: https://incredible-gumdrop-dcd0c5.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.11 Explain Closures and Use cases of Closures.

Ans:A closure is a feature of JavaScript that allows inner functions to access the outer scope of a function.In JavaScript, closure provides access to the outer scope of a function from inside the inner function, even after the outer function has closed.

Example:

```
// outer function
function greet() {
    let name = 'John';
    // inner function
    function displayName() {
          // accessing name variable
        return 'Hi' + ' ' + name;
     }

    return displayName;
}

const g1 = greet();
console.log(g1); // returns the function definition
console.log(g1()); // returns the value
```

Q.12 Create a Blog web app using JavaScript Fetch data from https://jsonplaceholder.typicode.com/posts and show it to ui.

Ans:

Github Link: https://github.com/gerard0lucas/blog-app-with-only-js-with-API

Deployed Link: https://effortless-cocada-1fba65.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

React Questions

Q.1 What's React and What are the advantages of it?

Ans:

The React.js framework is an open-source JavaScript framework and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript(plain javascript).

Advantages:

>It is very simple to use.
>The react components are reusable.
>Easy iteration of components.
>Accelerated Speed due to Virtual DOM

Q.2 What's Virtual Dom in React & What are the advantages of it?

Ans: Virtual DOM is a JavaScript representation of the actual DOM (the structure of your website or app). This JavaScript representation is like a blueprint that React uses to update the real DOM.React takes the virtual DOM and compares it to the previous virtual DOM to see what's changed. Once it knows what's changed, it updates the real DOM with the minimum amount of changes necessary.

Advantages:

>Fast and efficient, because React is only making the changes it needs to make, instead of redrawing the whole thing.
>Virtual DOM is simpler than most of the other approaches to making JavaScript reactive.
>The Virtual DOM makes optimised use of the memory compared to other systems because it doesn't hold observables in the memory

Q.3 Explain LifeCycle of React Components?

Ans: Each React Component goes through different stages of the lifecycle that are called Phases in React terminology. All React Component Lifecycle methods can be split into four phases of React Component.

1. Initialization
2. Mounting
3. Updating
4. Unmounting

The process in which all these phases are involved is called the **component's lifecycle**.

Q.4 What's the difference between between Functional Components and Class Components?

Ans: One of the main differences between functional and class components in React is their syntax. Functional components are written as plain JavaScript functions that take in props as an

argument and return a React element. On the other hand, class components require you to extend the React.Component class and define a render function that returns a React element.

Example for class component:

```
class Greetings extends React.Component {
render() {
return <h1>Welcome, {this.props.name
}</h1>;
}}
```

Example for functional component:

```
function Greetings(props) {
Return <h1>Welcome, {props.name}</h1>;
 }
```

Q.5 What are the hooks in React & Can we use Hooks in Class Components?

Ans: React Hooks are in-built functions that allow React developers to use state and lifecycle methods inside functional components, With React Hooks, developers get the power to use functional components for almost everything they need to do from just rendering UI to also handling state and you can also make a custom hooke and make use of it in react functional components.

You can't use a hook directly in the react class component.

Q.6 What are the LifeCycle methods and the advantages of it?

Ans: Lifecycle methods are special methods built into React, used to operate on components throughout their duration in the DOM. For example, when the component mounts, renders, updates, or unmounts. The most important lifecycle method, the render method.

There are four stages of LifeCycle method.

-Creation of the component (componentDidMount)
-Render of the component (render)
-Update of the component (componentDidUpdate)
-Death of the component (componentWillUnmount)

Advantages:

-It is very simple to use.
-The react components are reusable.
-Easy iteration of components.
-Accelerated Speed due to Virtual DOM

Q.7 What’s useState Hook & Advantages of it?

Ans: The useState() hook allows you to create, track and update a state in functional components. This hook is created by React and doesn't need to be installed when you create your React project.In React, the state is data or properties you can use in our application. States are mutable, meaning their value can change, and for that, the useState() hook is used to handle and manage your states.

Example for useState Hook:

```
const App = () => {
   const [count, setCount] = useState({ number: 5 });  //useState Hook
   return (
      <div>
         <p>{count.number}</p>
      </div>
   );
};
```

The advantage of React Hooks is that they allow developers to use state and other React features in functional components, rather than having to use class components.

Q.8 Explain useEffect & Advantages of it.

Ans: The useEffect hook is useful when you wish to run some functions during the component's lifecycle. For example, if you want to update the UI when a state changes, the useEffect hook is the way to go. You can also define a state on first load (when componentDidMount), and also clean the state when the component is unmounting (componentWillUnmount).
The advantage of Placing useEffect inside the component lets us access the count state variable (or any props) right from the effect.

Q.9 Explain Context Api and create a minor project on it, Create dashboard and with button on clicking on that change theme to dark and light

Ans:
Github Link: https://github.com/gerard0lucas/light-and-dark-theam-using-react
Deployed Link: https://master--starlit-biscotti-a7f3d2.netlify.app/

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.10 Explain useReducer and Its advantages.

Ans: The useReducer hook is another built-in hook in React that allows us to manage state in functional components. It is similar to the useState hook in that it takes an initial state value as an argument, but instead of returning just the current state value and a function to update it, it returns a state value and a dispatch function.The dispatch function takes an action object as an argument and returns a new state value based on that action.

Advantages:
-Manage all your state mutations in one place.
-Reducers are pure functions and very unit testable
-Reducers are handy for creating small pools of isolated data contexts. You can use it in conjunction with React Context API to keep your state where it belongs.

Q.10 Explain useReducer and Its advantages.

Ans:useReducer is a React hook which enables the use of this advanced pattern. When you call useReducer, you must pass the reducer (function) that will be used, and your initial state. It's important to remember that useReducer is generally used in place of useState in a component, because they're solving the same problem.
The big advantage of Reducers they are pure functions, and this means they have no side effects and must return the same outcome given the same arguments. It is easier to test them because they do not depend on React.

Q.11 build a Todo Web App Using React and useReducer Hook.

Ans:
Github Link:https://github.com/gerard0lucas/todo-with-use-reducer-in-react
Deployed Link: https://master--silver-jelly-9dc3c1.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.12 Build A simple counter app using React.

Ans:
Github Link: https://github.com/gerard0lucas/counter-app-using-react
Deployed Link: https://master--classy-creponne-c60087.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.13 Build Calculator Using React Only.

Ans:
Github Link: https://github.com/gerard0lucas/calculactor-app-with-react
Deployed Link: https://master--strong-pudding-a63de2.netlify.app/
Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.15 Explain Prop Drilling & How can we avoid it?

Ans: Prop drilling in react is the process of passing data from the parent component down to its children and then those children pass the same data down to their own children.
Sharing this data manually might be difficult, especially when it involves several nested components. Additionally, it may be difficult to share data between two child components. As a result, global state management is required.
It is not a good idea to pass data via several components when building neat, reusable code.
How to avoid it:
The React context API is a fast way of avoiding prop drilling and ensuring your data is managed globally.
You can also use third-party state management app like Redux.

Express Question

Q.1 Create a simple server using Express and connect with backend and create an endpoint “/post” which sends 20 posts

Ans:

Github Link: https://github.com/gerard0lucas/simple-express-server

Deployed Link: https://express-0hve.onrender.com/post

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.2 Explain a middleware and create a middleware that checks is user authenticated or not then send data of post.

Ans:

Github Link: https://github.com/gerard0lucas/Express-server-with-authentication

Deployed Link: https://express-server-with-authentication.onrender.com/post

Login details:

userId = admin

password = admin123

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.3 Create a backend for blog app, where user can perform crud operations.

Github Link: https://github.com/gerard0lucas/Backend-crud-

Deployed Link: https://backend-curd-with-express.onrender.com/api/blogs

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.4 What is the difference between authentication and authorization?

Ans: Authentication is the process of verifying the identity of a user.Authentication is the first step in the access control process. It ensures that the user is able to access the system.

Authorization process determines what a user is allowed to do on an application. It involves granting access to resources based on the user’s identity and their permissions. within the application. Authorization usually follows authentication.

Q.5 What is the difference between common JS and EJS modules?

Ans:

CommonJS is a module system inspired by the AMD API from Microsoft. It allows. modules to be written in a similar way to AMD modules, with the main difference being that CommonJS modules can be loaded into any node.js environment.

ES Modules are a new module system developed by Google which allows for greater modularity and reuse across different programming languages. The advantage of using ES Modules is that They can be compiled into native code which makes them faster and more efficient than CommonJS modules.

Q.6 What is JWT and what we can achieve with that create a minor project with Jwt.

Ans: JWT stands for **JSON Web Token**, is an open standard used to share security information between two parties, client and server. Each JWT contains encoded JSON objects, including a set of claims. JWTs are signed using a cryptographic algorithm to ensure that the claims cannot be altered after the token is issued.

JWTs are a good way of securely transmitting information between partiesWTs are a good way of securely transmitting information between parties

Github Link: https://github.com/gerard0lucas/Workout-buddy

Deployed Link: https://work-bxsi.onrender.com/api/workouts

Note: The links may perform slower than usual because I hv used free hosting services for deployment.

Q.7 What should we do with the password of a user before storing it into DB?

Ans: When a user input a password, in the backend its hash value is calculated and to provide security against password and salt is also added while calculating the hash value and then this hash value is stored in the database.

Hash values are stored in the database because these hash algorithms are based on one way functions which given a string it is easy to calculate the hash value but given the hash value it's very hard to find the string.

For example:

The SHA256 hash of "12345" is "5994471abb01112afcc18159f6cc74b4f511b99806da59b3caf5a9c173cacfc5" . If this hash is seen in a database, we know that the user's password is "12345".

Q.8 What is an event loop in NodeJS ?

Ans: The event loop allows Node.js to perform non-blocking I/O operations despite the fact that JavaScript is single-threaded. It is done by assigning operations to the operating system whenever and wherever possible.

An event loop is an endless loop, which waits for tasks, executes them, and then sleeps until it receives more tasks.It executes the tasks starting from the oldest first.

Q.9 Create a Full Stack Ecommerce website with all major functionalities.

Ans:

backend:

Github Link: https://github.com/gerard0lucas/ecom-server

Deployed Link: https://ecomserver-pcxl.onrender.com/api/user/get

Frontend:

Github Link: https://github.com/gerard0lucas/ecom-client

Deployed Link: https://master--unique-trifle-00202c.netlify.app/