Web Developer Intern Assessment

https://github.com/Deepthi875/Web-Development-

Section 1: JavaScript [Mandatory]

Conceptual Question:

1. == (Loose Equality):

== checks if two values are equal, but it doesn't care if the types (like string or number) are different.

If the types are different, JavaScript tries to convert one to match the other, then it checks if they are equal.

2. === (Strict Equality):

=== checks if two values are the same in both value and type.

It doesn't try to convert anything. So, if the types are different (like number vs. string), it will return false, even if the values look the same.

Section 2: React [Mandatory]

Conceptual Question:

What is the purpose of React's `useEffect` hook? Provide an example use case

The useEffect hook in React is used to run some code after the component appears on the screen. It's often used to do things like:

- Fetch data from a server
- Update something outside React, like the page title
- Set up things like timers or event listeners
- It runs after the component renders (shows up on the screen).
- You can also make it run only once when the component first loads by giving it an empty list [] as the second argument.

```
import React, { useState, useEffect } from 'react';

function DataFetchingComponent() {
  const [data, setData] = useState([]); // Start with an empty list

  useEffect(() => {
    // Code here runs once when the component loads
    fetch('https://api.example.com/data')
      .then(response => response.json())
      .then(data => setData(data)); // Store the data we get in 'data' state
  }, []); // Empty array means run once
```

```
  return (
    <div>
      <h1>Fetched Data:</h1>
      <ul>
        {data.map((item) => (
          <li key={item.id}>{item.name}</li> // Show each item in a list
        ))}
      </ul>
    </div>
  );
}

export default DataFetchingComponent;
```

Section 3: Node.js [Mandatory]

Conceptual Question

Node.js handles asynchronous operations by using something called non-blocking programming. This means Node.js can start an operation (like reading a file or fetching data from a database) and then move on to other tasks instead of waiting for the operation to finish. When the operation is done, Node.js comes back to it and processes the result.

- Node.js uses an event loop to manage tasks.
- When you start a task that takes time (like fetching data from an API), Node.js "schedules" it and then keeps running other code.
- Once the task finishes, Node.js uses a callback or promise to handle the result.

Efficiency: Node.js can handle many tasks at the same time without slowing down. For example, while waiting for data from a database, it can handle other requests.

Scalability: This makes Node.js great for building applications that need to handle many users at once, like chat apps or APIs.

Section 4: Next.js [Mandatory]

Conceptual Question:

In Next.js, both getStaticProps and getServerSideProps are used to fetch data, but they work differently in terms of when and how the data is fetched.

getStaticProps

Runs only once at build time, meaning when you build your app.

It fetches data ahead of time, creates a static version of the page, and then serves that same page to all users.

Content that doesn't change often, like blog posts or product pages. It's super fast because the data is already ready and doesn't need to be fetched on each request.

Example use case: A homepage that displays a list of items that won't change often.

getServerSideProps

Runs on every request, meaning each time a user visits the page.

It fetches fresh data each time the page loads, so it's always up-to-date with the latest information.

Content that changes frequently, like user-specific data, a dashboard, or real-time information. This method is slower than getStaticProps because it fetches data each time a user visits.

Example use case: A dashboard that shows the latest user data or a stock prices page.

Section 5: PostgreSQL [Mandatory]

Conceptual Question:

A primary key in PostgreSQL is a special column (or a set of columns) in a table that makes sure each row is unique. No two rows can have the same value in the primary key column.

- Uniqueness: The primary key ensures that every row in the table is different from every other row. For example, if you have a users table, each user must have a unique id.
- Faster Searches: PostgreSQL automatically creates an index on the primary key, which makes searching for specific data faster.
- Data Consistency: The primary key helps keep the data organized and prevents mistakes, like having two users with the same id.
- Relationships Between Tables: A primary key is often used in other tables to create relationships. For example, if another table needs to know which user a post belongs to, it will use the user's id as a reference.

Section 6: Azure App Services [Mandatory]

1. Conceptual Questions:

Prepare Your Node.js Application:

- Ensure your application has a package.json file and a start script
- Test your application locally to ensure it's working as expected.

Create an Azure Account and Set Up Azure App Service:

- Sign in to the Azure Portal.

- Go to App Services and click Create to start creating a new App Service.

Configure Your App Service:

- Select the Subscription and Resource Group (create a new one if necessary).
- Choose an App name (which will be your app's URL, e.g.,
- Select the Runtime stack as Node.js and the appropriate version for your application.
- Choose a Region close to your target audience.

Deploy the Application:

- Use Azure CLI, GitHub Actions, or VS Code for deployment.

o For example, using the Azure CLI:

1. Open a terminal and sign in with az login.
2. Deploy your code using the command:bash

Access Your App:

- Once deployed, you can access your app at https://yourappname.azurewebsites.net.

Common Benefit of Using Azure App Services for Hosting Applications

A key benefit of Azure App Services is automatic scaling and management. This means Azure can automatically handle increases in traffic by scaling up resources as needed, ensuring reliable performance without manual intervention. Azure App Services also offers features like continuous deployment, security management, and integration with other Azure services, which helps simplify the hosting and management of applications.

Section 7: WordPress - Basic Custom Plugin

Conceptual Question:

In WordPress, posts and pages are used for different types of content. Posts are meant for things like blog articles, news updates, or any content that is updated regularly. They show up in reverse order, with the newest post first, and are organized by date. Posts can be grouped into categories and tagged with keywords to make them easy to find. They also go out in RSS feeds, which lets people subscribe to get updates, and usually have a comment section so readers can leave feedback.

Pages, on the other hand, are used for more permanent content, like an “About Us” or “Contact” page. They don’t follow any timeline, and they aren’t included in the main blog feed. Pages can be arranged in a hierarchy, so you can make a page a “parent” or “child” of another, which helps

keep things organized. Pages don’t use categories or tags, and they’re not part of RSS feeds because they don’t need to be updated often. In short, posts are best for regularly updated content, while pages are great for information that doesn’t change much over time.