**Part 2: General Programming Questions**

Algorithmic Problem:

       ● Implement a function in a language of your choice that checks if a

given string is a palindrome.

**Solution:**

def is\_palindrome(s):

    s = ''.join(filter(str.isalnum, s.lower()))

    start, end = 0, len(s) - 1

    while start < end:

        if s[start] != s[end]:

            return False

        start += 1

        end -= 1

    return True

user\_input = input("Enter a string: ")

result = is\_palindrome(user\_input)

if result:

    print("The given string is a palindrome.")

else:

    print("The given string is not a palindrome.")

            Database Concepts :

       ●    Explain the difference between SQL and NoSQL databases. Provide

examples of use cases for each.

**Solution:**

SQL databases are relational databases, meaning they store data in tables with rows and columns. NoSQL databases are non-relational databases, meaning they don't use tables.

SQL databases are better suited for storing structured data, while NoSQL databases are better suited for storing unstructured data. Structured data is data that has a defined format, such as a customer's name and address. Unstructured data is data that doesn't have a defined format, such as a customer's review of a product.

SQL databases are better for complex queries, while NoSQL databases are better for real-time data processing. Complex queries are queries that involve multiple tables and joins. Real-time data processing is the process of processing data as it's generated.

Here are some examples of use cases for SQL and NoSQL databases:

**SQL databases:**

Online transaction processing (OLTP) systems

Customer relationship management (CRM) systems

Enterprise resource planning (ERP) systems

**NoSQL databases:**

Big data processing

Real-time analytics

Content management systems (CMS)

E-commerce platforms

Web Technologies :

                  ●    What is CORS, and how does it work? How can you handle

CORS-related issues in a web application?

**Solution:**

Cross-Origin Resource Sharing (CORS) is an HTTP mechanism that allows resources to be shared securely between origins. It relaxes the same-origin policy of modern browsers.

CORS works by adding HTTP headers to requests and responses. These headers indicate whether the request or response is allowed to access resources. For example, a server configured for CORS will return an extra header with “Access-Control-Allow-Origin” on each response.

It allows restricted resources on a web page to be accessed from another domain. These resources may include: Images, Stylesheets, Scripts, Iframes, Videos, Web fonts.

CORS errors occur when a webpage makes a request to a different domain than the one that served the page. The server responds with an HTTP error because the “Origin” header in the request is not allowed by the server's CORS configuration.

**To troubleshoot CORS errors, you can:**

Check the browser console for CORS errors

Verify that the server is sending the correct CORS headers

Ensure that a firewall or proxy does not block the request

Check if the request is being made using HTTPS or HTTP

**To handle CORS errors, you can:**

Fix server-side configuration

Use a proxy server

Use JSONP (JSON with Padding)

Use CORS-anywhere

Coding Best Practices:

       ●    Discuss three coding best practices that you consider important for

maintaining a clean and efficient codebase.

**Solution:**

1. Use meaningful names: Give meaningful names to variables, classes, methods, and other entities in the code.
2. Write readable code: Make your code easy to read and understand.
3. Use consistent formatting: Use consistent line breaks, space, and indentation.