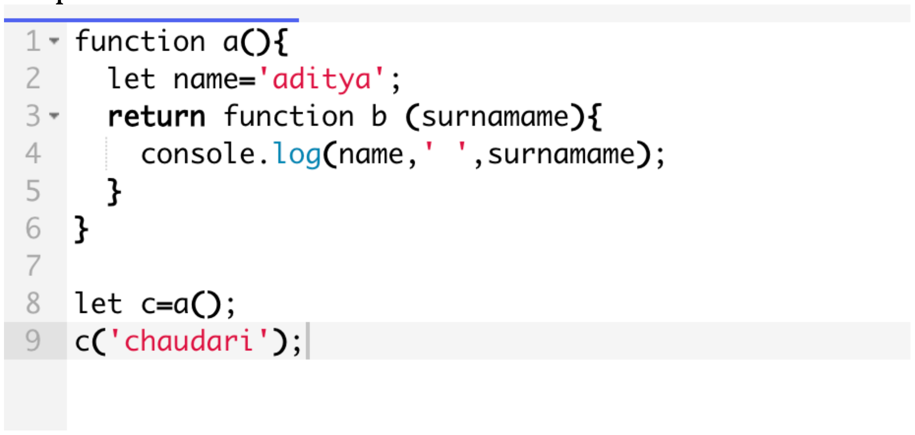
Front End Questions

1. Lifecycle of Angular
2. How to pass data between angular components
3. What is NgModule – Comilation context for Services,Component, How they related to each other
4. What is Observable
5. Difference between Observable and Promise
6. Explain SOLID principles with example in Angular
7. How to enable JIT always - `angular.json aot = false`
8. How to increase performance of javascript code
   `Bundling, Async Programing, DOM Manipulation`

9. How to increase Performance of Angular App.
10. How to design a layout using CSS : Header,Navigations,Main Content, Side Bar, Footer
11. Explain Javascript Closures
    Closures allow functions to retain access to variables from their lexical scope
    

```
function a(){
  let name='aditya';
  return function b (surnamame){
    console.log(name,' ',surnamame);
  }
}

let c=a();
c('chaudari');
```

12. How a Javascript page in loaded
    1.Browser Requests the Page:
    2. Server Responds:
    The server processes the request and sends back an HTTP response containing the HTML, CSS, and JavaScript files.
    3. Browser Parses the HTML:
    The browser starts parsing the HTML document from top to bottom.
    As it encounters different tags, it processes them accordingly:
    HTML tags: Structure and content of the page.
    CSS links or styles: Apply styles to the HTML elements.
    JavaScript: Execute scripts that add functionality to the page.

13. How HTML page loads

1.Browser Requests the Page:
2. Server Responds:
The server processes the request and sends back an HTTP response containing the HTML, CSS, and JavaScript files.
3. Browser Parses the HTML:
The browser starts parsing the HTML document from top to bottom.
As it encounters different tags, it processes them accordingly:
HTML tags: Structure and content of the page.
CSS links or styles: Apply styles to the HTML elements.
JavaScript: Execute scripts that add functionality to the page.
4. Javascript Loading
5. DOM Initialization
6.Rendering

14. State management in angular

Using NgRx for State Management

NgRx is a powerful state management library for Angular applications that implements the Redux pattern. It helps manage the state of the application in a reactive way using observables.

Key Concepts in NgRx

Store: The single source of truth for the application state.

Actions: Plain objects that describe an event that has occurred (e.g., user login, data fetch).

Reducers: Pure functions that take the current state and an action, and return a new state.

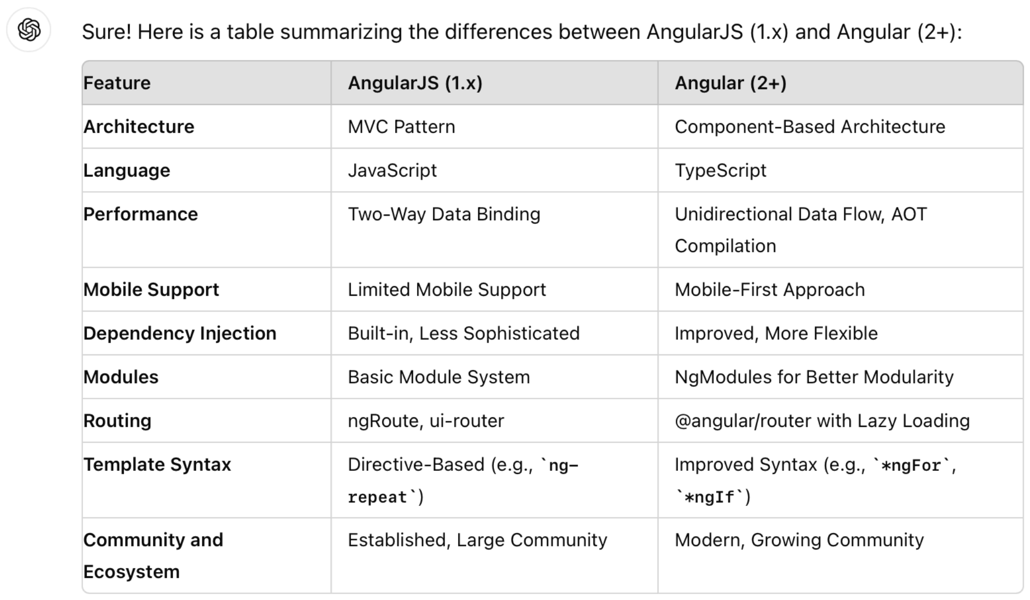
Selectors: Functions used to select, derive, and compose pieces of state.

Effects: Side-effects that interact with external services and dispatch new actions.

15. Explain how the angular app executes and how the browser knows the running app is angular app .
    - Index.HTML (contain Bundle)
      - index HTML contain scripts, polyfils
      - Loading Javascript File

- loading Bootstraping Module
- Rendering Component
- Loading App

16. What is the difference between AngularJs and Angular 2+



Sure! Here is a table summarizing the differences between AngularJS (1.x) and Angular (2+):

Feature	AngularJS (1.x)	Angular (2+)
Architecture	MVC Pattern	Component-Based Architecture
Language	JavaScript	TypeScript
Performance	Two-Way Data Binding	Unidirectional Data Flow, AOT Compilation
Mobile Support	Limited Mobile Support	Mobile-First Approach
Dependency Injection	Built-in, Less Sophisticated	Improved, More Flexible
Modules	Basic Module System	NgModules for Better Modularity
Routing	ngRoute, ui-router	@angular/router with Lazy Loading
Template Syntax	Directive-Based (e.g., `` `ng-repeat` ``)	Improved Syntax (e.g., `` `*ngFor` ``, `` `*ngIf` ``)
Community and Ecosystem	Established, Large Community	Modern, Growing Community

17. Is there any way in Angular where users do not need to subscribe and unsubscribe from the observable?

    `async` Pipe, `takeUntil` with `Subject`, **NgRx `ComponentStore`**:

18. What are commonly used attributes in the @Component decorator
    selector,templateUrls,styleUrls

What is an Interceptor? Why was the interceptor needed?
How intercept Works

Clone and Modify Request: You can modify the outgoing request (e.g., add headers, change the URL) by cloning it and applying changes.

Pass to Next Handler: You pass the modified request to the next handler. This allows the request to continue through the interceptor chain or be sent to the backend if no other interceptors are present.

Handle Response or Errors: You can handle responses and errors using RxJS operators such as catchError. This allows you to perform actions such as logging errors or retrying requests.

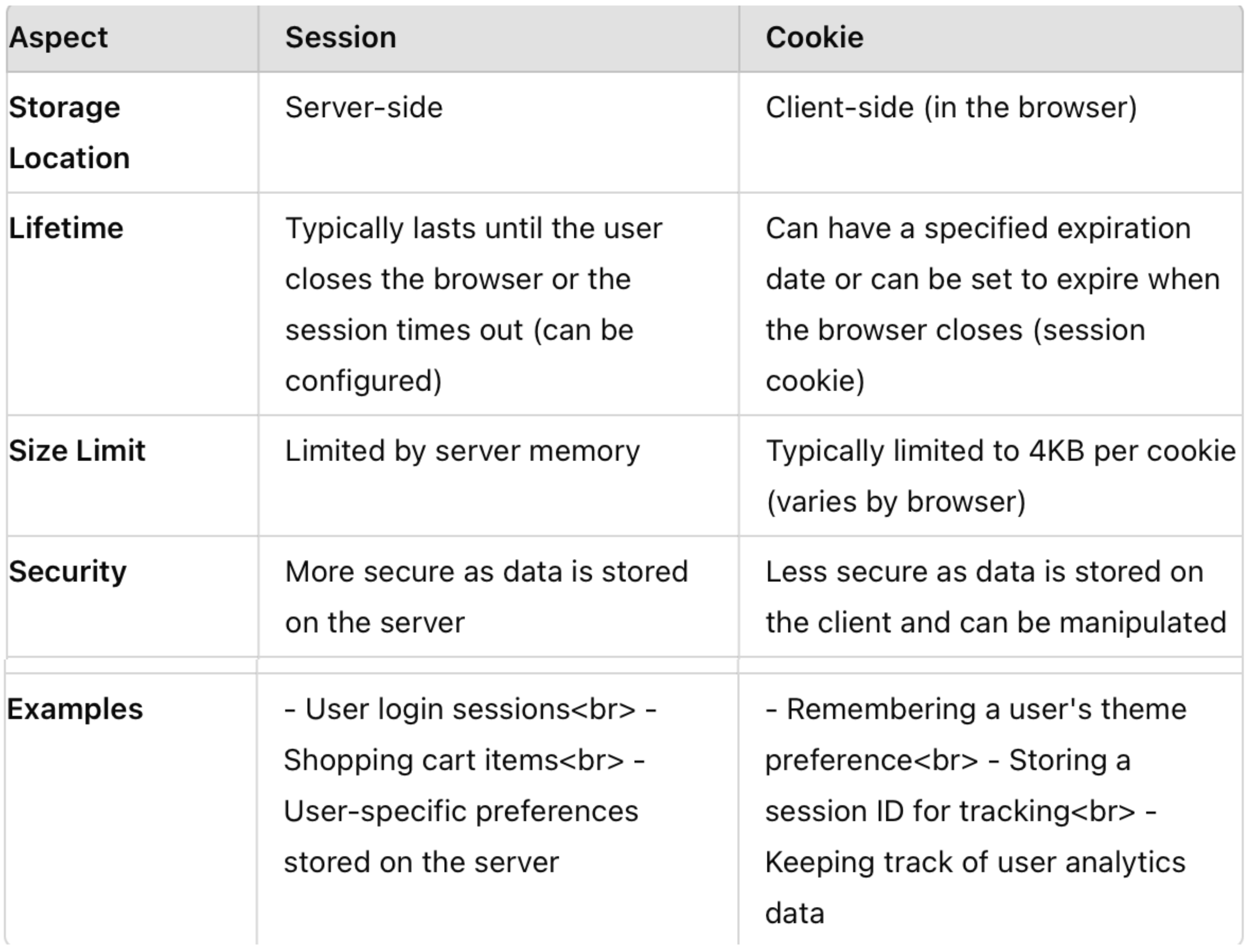
19. What is AuthGuard? why it needed
20. Why are services required in angular? How to communicate with API using services.
21. Explain Authentication Services? and where to store auth token?
22. Did you write unit test cases in angular?
    Yes. By Jasmin Framework run on Karma,
    We used to do that in Spec file by TestBed import

23. What is the output of below Javascript code

```
console.log('Start');
 setTimeout(() => {
       console.log('setTimeout callback');
}, 0);

 Promise.resolve().then(() => {
      console.log('Promise resolved');
});

console.log('End');
```

Backend Questions :

24. OPEN Closed Principles
25. SOLID Principles
26. Explain Parking ticketing system design.
27. Unit test case of service layer in C#
    Explain XUnit , getUser, Arrange, Act and Assert
28. Explain token authentication.
29. Difference between Session and cookie



Aspect	Session	Cookie
Storage Location	Server-side	Client-side (in the browser)
Lifetime	Typically lasts until the user closes the browser or the session times out (can be configured)	Can have a specified expiration date or can be set to expire when the browser closes (session cookie)
Size Limit	Limited by server memory	Typically limited to 4KB per cookie (varies by browser)
Security	More secure as data is stored on the server	Less secure as data is stored on the client and can be manipulated
Examples	- User login sessions - Shopping cart items - User-specific preferences stored on the server	- Remembering a user's theme preference - Storing a session ID for tracking - Keeping track of user analytics data

30. Explain Repository Pattern

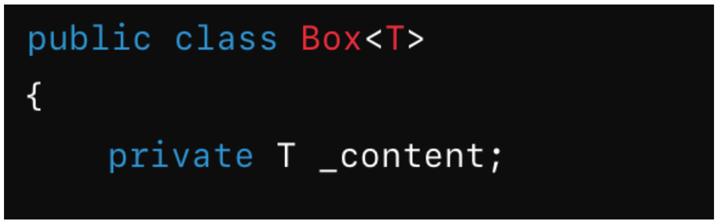
The Repository Design Pattern provides a **clean separation of concerns**, enhances testability, and makes the codebase more maintainable by abstracting data access logic. In the provided example, it helps in managing products in a **product management system by providing a consistent way to access data**.

Step-by-Step Implementation:

1. **Define Entity Classes**: These represent the data model.
2. **Create the Repository Interface**: This defines the methods for data access.
3. **Implement the Repository Interface**: The concrete class that handles the actual data access logic.
4. **Use Dependency Injection**: Inject the repository into the services or controllers where it is needed.

31. Explain how dot net core app executes.
32. What is Generics in c# with example?

Generics in C# allow you to define classes, interfaces, and methods with a placeholder for the data type. This enables you to create more flexible and reusable code while maintaining type safety. By using generics, you can write code that works with any data type without sacrificing performance or type safety.



```
public class Box<T>
{
    private T _content;
```

33. Difference between .NET core 3.0 and .NET core 6.0?
34. Which testing framework is used in your .NET.

Coding Challenge

35. Find characters with maximum occurrences.
    a. input: char[] ={"a","e","c","b","d","d","c","b","d","a","c","d"}
    b. Output =[4,3,2]

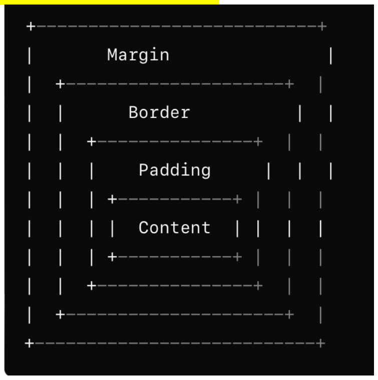
```
class Program
{
    static void Main()
    {
        // Input character array
        char[] input = { 'a', 'e', 'c', 'b', 'd', 'd', 'c', 'b', 'd', 'a', 'c', 'd' };

        // Find characters with the maximum occurrences
        List<int> result = FindMaxOccurrences(input);

        // Display the result
        Console.WriteLine("[" + string.Join(",", result) + "]");
    }

    static List<int> FindMaxOccurrences(char[] array)
    {
        // Dictionary to store frequency of each character
        Dictionary<char, int> frequencyMap = new Dictionary<char, int>();

        // Populate the frequency map
        foreach (char ch in array)
        {
            if (frequencyMap.ContainsKey(ch))
            {
                frequencyMap[ch]++;
            }
            else
            {
                frequencyMap[ch] = 1;
            }
        }

        // Get the list of frequency values and sort them in descending order
        List<int> sortedFrequencies = frequencyMap.Values.OrderByDescending(count =>
count).ToList();

        return sortedFrequencies;
    }
}
```

36. Write a C# code to swap string in following order
    Need to swap only 2 character with next 2 character
    Input: Siddique Output: ddsiueiq

```

public class HelloWorld
{
    public static void Main(string[] args)
    {
          // Input string
        string input = "Siddique";

        // Get the swapped string
        string output = SwapEveryTwoCharacters(input);

        // Display the result
        Console.WriteLine(output);  // Output: ddsiueiq
    }

    static string SwapEveryTwoCharacters(string input)
    {
        // Convert input string to a character array for manipulation
        char[] chars = input.ToCharArray();

        // Iterate through the string in chunks of four characters
        for (int i = 0; i < chars.Length - 3; i += 4)
        {
            // Swap the first two characters with the next two characters within each
chunk
            Swap(chars, i, i + 2);
            Swap(chars, i + 1, i + 3);
        }
        // Convert the character array back to a string
        return new string(chars);
    }

    static void Swap(char[] array, int index1, int index2)
    {
        char temp = array[index1];
        array[index1] = array[index2];
        array[index2] = temp;
    }
}
```

Frontend questions

1. What is the difference between let, var,
2. What are closures?
3. What is the difference between promise and observable?
4. Can we unsubscribe observables? **Yes, by .UnSubscribe() and TakeUntil(OnDestroy)**
5. Can we unsubscribe promise? **No, you do not unsubscribe from Promises.**
6. What is the pipe and map RxJS?
7. What is an interceptor? Why do we need an interceptor?
8. What is box css?



```
+---------------------------+
|      Margin               |
|  +---------------------+  |
|  |     Border          |  |
|  |  +---------------+  |  |
|  |  |   Padding     |  |  |
|  |  | +-----------+ |  |  |
|  |  | |  Content  | |  |  |
|  |  | +-----------+ |  |  |
|  |  +---------------+  |  |
|  +---------------------+  |
+---------------------------+
```

9. explain the position in css? > Static , Relative, Absolute or Fixed
10. explain the module in angular?
11. How does angular work?
12. What is the output of flowing code
13. What is the extension method?
14. What is the difference between setTimeOut and setInterval?
15. Which is the entry point of angular?

16. Write a code to get multiplication of numbers? also explain what is the name of this feature? mul(2,3) && mul(2)(3)

    Function currying

    public static Func<int, int> Mul(int a) { return b => a * b; }

// Currying multiplication Func<int, int> curriedMul = Mul(2); int result2 = curriedMul(3); Console.WriteLine($"Mul(2)(3) = {result2}");

17. What is the output of flowing code

if {
var a = 10;
let b = 20;
}
console.log(a);
console.log(b);
Ans – 10, undefined
—---------------------------
var a = 2;
let b = 5;
if {
var a = 10;
let b = 20;
}
console.log(a);
console.log(b);
Output = 10,5
—---------------------------
var obj = {name="raj", surname="sharma"}
var obj2 = obj;

obj2.name = "test";

console.log(obj.name);
console.log(obj2.name);
Output = "test","test" because it is refering to same object value

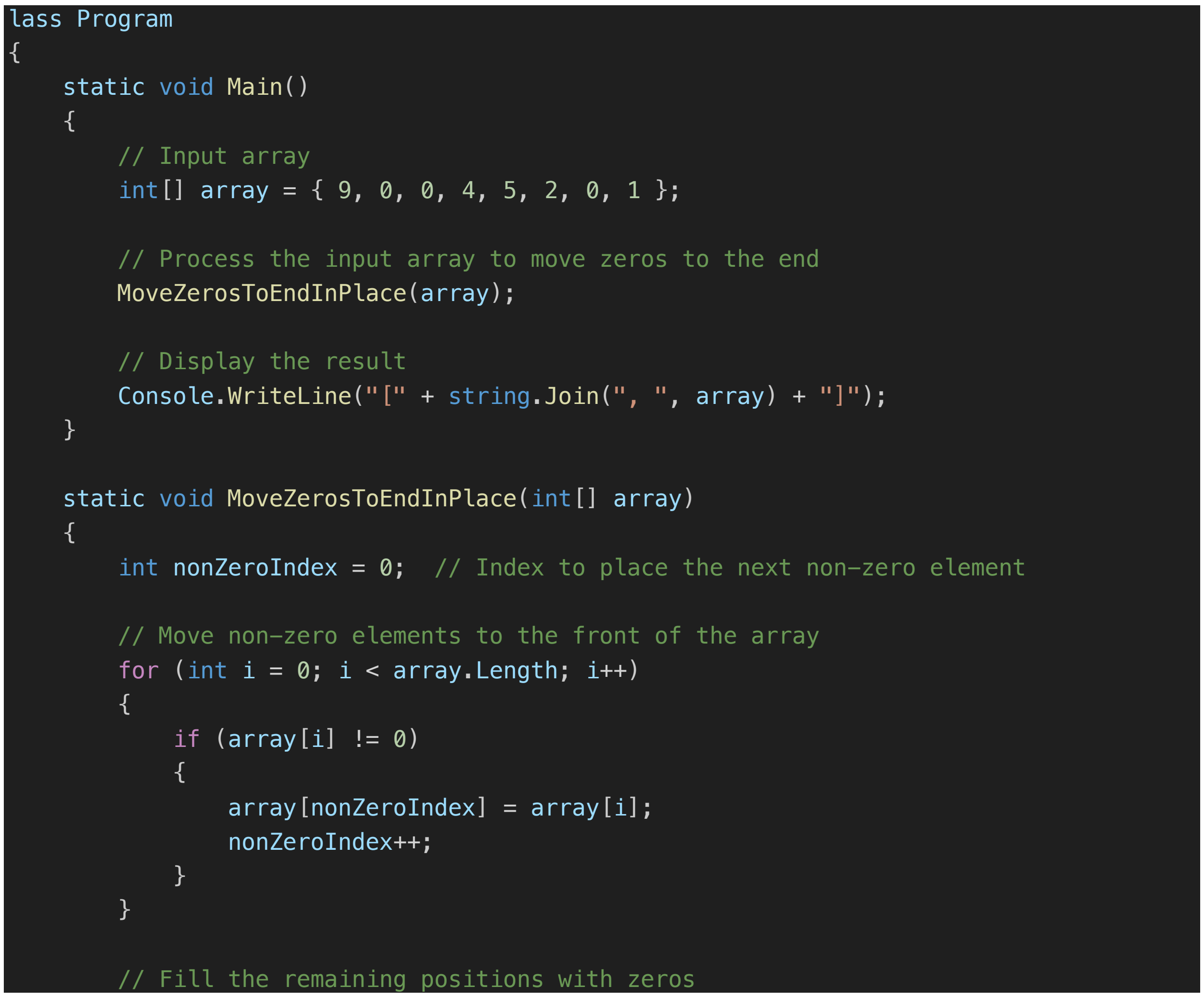
Backend questions

1. What is middleware?
2. What is a short circuit?
3. How to secure web api?
4. Have you implemented JWT?
5. What is multicast delegate
   Multicast delegates are a powerful feature in C# that allow you to aggregate multiple method calls into a single delegate instance. This is particularly useful for event handling, where you often need to notify multiple subscribers about an event. By using multicast delegates, you can simplify the management of method invocations and ensure that all relevant actions are performed in response to a single trigger.

6. What is the difference between func and predicate?
7. What are the components of JWT?

8. Suppose If we only declare methods in abstract class, then what is the difference between abstract class and interface?
9. What is a kestrel?
10. How does the kestrel server work in deployment?
11. What is encapsulation?
12. What is method hiding?
13. What is async and await?
14. Can the compiler execute code with async and without await? And vice versa?

15. Write a program to get desired output,
    Input: [9,0,0,4,5,2,0,1]
    Output: [9,4,5,2,1,0,0,0]



```
lass Program
{
    static void Main()
    {
        // Input array
        int[] array = { 9, 0, 0, 4, 5, 2, 0, 1 };

        // Process the input array to move zeros to the end
        MoveZerosToEndInPlace(array);

        // Display the result
        Console.WriteLine("[" + string.Join(", ", array) + "]");
    }

    static void MoveZerosToEndInPlace(int[] array)
    {
        int nonZeroIndex = 0;  // Index to place the next non-zero element

        // Move non-zero elements to the front of the array
        for (int i = 0; i < array.Length; i++)
        {
            if (array[i] != 0)
            {
                array[nonZeroIndex] = array[i];
                nonZeroIndex++;
            }
        }

        // Fill the remaining positions with zeros
        for (int i = nonZeroIndex; i < array.Length; i++)
        {
            array[i] = 0;
        }
    }
}
```

16. Can we get above program output without using a second array?