**How he/she solved the codility test –**

understand his/her logic.

Clarity in problem understanding.

Basic problem-solving attitude.

Clear fundamentals

time and space complexity

optimize the time or space complexity

different solution

why this technology

Can you again write clean, efficient, and maintainable code

handle errors

logs

identify potential sources of errors in a given piece of code

balance between readability and performance

optimized your code

what would you do differently if you were to start this project again?

**Your Contribution on the Academic Project –**

Can you provide an overview of your academic project?

What was the primary goal or objective of the project? What problem were you trying to solve with this project?

Can you discuss any challenges you faced during the implementation phase and how you overcame them?

Were there any specific techniques, algorithms, or tools you employed during the project?

What specific role did you play in the project?

Can you describe the tasks or responsibilities you were assigned?

What skills or knowledge did you gain from working on this project?

How do you think this project has contributed to your overall academic or professional development?

Do you have any ideas for potential future improvements or extensions to the project?

what would you do differently if you were to start this project again?

**Object oriented concepts  -**

**Testing and Mocking:**

memory allocation?

garbage collection

memory management

design pattern

Are you familiar with SOLID principles? Can you explain each one?

What is an interface, and how does it differ from a class?

**Method Overloading vs. Method Overriding:**

What is the difference between a class and an object?

What is abstraction,

Explain the concept of polymorphism and its different forms (compile-time and runtime).

What is inheritance, and how does it facilitate code reuse?

Provide an example of how you've used encapsulation

**Elementary geometry/Mathematics –**

**Data Structures –**

different types of data structures

array- contiguous memory allocation- insertion, deletion, search

linked lists- reverse, doubly, singly

stacks and queues- LIFO and FIFO

tree- binary search tree, preorder, inorder, and postorder, balanced tree. find the height

graph- depth-first search, breadth-first search

hash table-

priority queue and a regular queue. -priority queue using a heap

**Round02**

**Data Structures –**

**Virtual tables –**

virtual tables and virtual functions.

structure and purpose of a virtual table?

How are virtual tables used in implementing polymorphism in languages like C++ and Java

How is a virtual table typically implemented in memory?

What happens when a class defines virtual functions?

Can you explain the role of function pointers in virtual tables?

How does inheritance affect virtual tables?

**STL –**

Standard Template Library

fundamental components of the STL.

What are containers in the STL? Name some commonly used container classes.

vector vs. a list vs. a deque.

Provide examples of common algorithms available in the STL.

Explain the concept of iterator categories (e.g., input iterators, output iterators, forward iterators, random access iterators).

Can you describe any design patterns or best practices related to using the STL effectively?

**Design and architecting  -**

Design principles and patterns

Discuss different architectural styles (e.g., monolithic, microservices, layered) and their respective advantages and disadvantages.

How do you approach designing a distributed system to ensure fault tolerance and high availability?

optimize the performance of a database-driven application

Can you describe any caching strategies you've implemented to improve performance in your projects?

Discuss techniques for mitigating common security threats such as SQL injection and cross-site scripting (XSS).

How do you ensure data integrity and confidentiality in a distributed system?

Can you describe any measures you've taken to ensure the reliability and fault tolerance of your systems?

Software development lifecycle

influencing software design and architecture in the near future?

How do you stay updated with the latest developments in software architecture and design

impact of cloud computing and serverless architecture on modern software systems.

**Debugging  -**

**Knowledge about assemblies - reference resolution –**

What are private and shared assemblies, and how do they differ?

Discuss the concept of assembly probing and its relevance during runtime.

How does the Common Language Runtime (CLR) locate and load assemblies?

Explain the process of assembly loading in the .NET runtime.

How does the assembly manifest contribute to versioning and deployment?

Can you describe how strong naming works and its role in ensuring assembly integrity?

Explain the difference between a DLL (Dynamic Link Library) and an EXE (Executable) assembly.

What is reflection, and how can it be used to inspect and manipulate assemblies at runtime?

Can you describe how to dynamically load and execute assemblies using reflection?

Discuss any security considerations when using reflection to interact with assemblies.

Can you describe any techniques for reducing the size of assemblies to improve deployment performance?

Discuss strategies for optimizing assembly loading and execution times.

Discuss techniques for diagnosing runtime errors or exceptions related to assemblies.

Can you describe how to dynamically load and execute assemblies using reflection?

What is the Global Assembly Cache (GAC), and when would you use it?

**Collections and delegates  -**

What are collections in .NET

difference between an array and a collection.

What is a List<T>,

differences between ArrayList and List<T>.

What is a Dictionary<TKey, TValue>, and how does it differ from other collection types?

What is the difference between a List<T> and a LinkedList<T>, and when would you choose one over the other?

What is a delegate,

Explain the similarities and differences between delegates and interfaces.

How do you declare and instantiate a delegate in C#?

delegates used in event handling in .NET?

Explain the difference between events and delegates.

Discuss the concept of anonymous methods and lambda expressions in relation to delegates.

ow would you implement a callback mechanism using delegates?

hat is the difference between synchronous and asynchronous delegates?

**ADO.net –**

Can you explain what ADO.NET is and its primary purpose

Differentiate between DataSet and DataReader.

different data providers available in ADO.NET?

the SqlConnection class versus OleDbConnection class?

Can you walk me through the process of connecting to a database and executing a query using ADO.NET?

What are SqlCommand and SqlParameter, and how are they used?

How would you handle parameterized queries to prevent SQL injection attacks?

What is a transaction, and how would you implement it using ADO.NET?

Explain the role of the DataAdapter in data binding.

How do you handle exceptions in ADO.NET?

How can you minimize round trips to the database?

Can you discuss the usage of stored procedures and user-defined functions in ADO.NET?

**Database Normalization –**

hat is database normalization, and why is it important?

Can you list and explain the different normalization forms (1NF, 2NF, 3NF, BCNF, 4NF, 5NF)?

Provide an example of a table that is not in 1NF and normalize it.

Explain the concept of partial dependency and how it relates to normalization.

Discuss the role of transitive dependency in normalization.

How does normalization affect database performance?

How does normalization impact query performance and optimization?

**Joins  -**

What is a SQL join, and why is it used?

Can you explain the difference between an inner join and an outer join?

Describe the different types of joins in SQL.

ow can you optimize join performance in SQL queries?

How do SQL joins handle NULL values in the joined columns?

**Functions Procedures Triggers –**

What is a database function, and what purpose does it serve?

Discuss the advantages of using stored procedures over inline SQL queries.

Describe the difference between DML triggers and DDL triggers.

When would you use a function over a stored procedure?

In what scenarios would you choose to use a trigger?

How do you call a stored procedure from within another stored procedure?

How do you handle return values from functions and stored procedures?

How do you implement error handling in triggers?

How can you optimize the performance of functions and stored procedures?

Discuss security considerations related to functions, stored procedures, and triggers.

What techniques do you use for debugging functions, stored procedures, and triggers?

Explain how transactions are managed within functions, stored procedures, and triggers.

When is it appropriate to use dynamic SQL within functions, stored procedures, or triggers?

What are the potential risks associated with dynamic SQL, and how can they be mitigated?

**Memory management –**

Explain the difference between stack and heap memory.

How does dynamic memory allocation work in languages like C/C++?

Can you discuss the difference between malloc() and new in C++?

What are the common methods of deallocating memory?

Explain the importance of properly deallocating memory to prevent memory leaks.

What is a memory leak, and how can it occur in a program?

Discuss strategies for detecting and fixing memory leaks in software.

What is memory fragmentation, and how can it impact system performance?

What is garbage collection, and in what programming languages is it commonly used?

Discuss techniques such as memory pooling, object pooling, and memory reuse.

What tools or techniques would you use to analyze memory usage in a program?

What is memory safety, and why is it important in programming?

Discuss common memory-related vulnerabilities, such as buffer overflows and dangling pointers.

How does memory management differ in single-threaded versus multithreaded environments?

What is virtual memory, and how does it work?

Discuss the benefits and drawbacks of using virtual memory in modern operating systems.

How does memory management differ in embedded systems compared to desktop or server environments?

What considerations should be taken into account when designing memory management systems for embedded devices?

**Bitwise Operations ­–**

an you explain the bitwise AND, OR, XOR, and NOT operations?

How are bitwise operations different from logical operations?

What is the significance of shifting operations (left shift, right shift) in bitwise operations?

How would you implement basic arithmetic operations (addition, subtraction, multiplication, division) using bitwise operators?

How do bitwise operations compare to arithmetic operations in terms of performance?

Given a binary representation of a number, how would you swap the values of two specific bits?

**Swap Numbers with Pointers –**

Can you explain the concept of pointer dereferencing?

Can you explain how you would swap the values of two variables without using a temporary variable?

Can you provide a code example of swapping two numbers using pointers in your preferred programming language?

What precautions should be taken when manipulating memory using pointers?

What happens if you attempt to dereference an uninitialized pointer?

Are there any performance implications of swapping numbers using pointers compared to other methods?